FACULDADE DE INFORMÁTICA

PUCRS - Brazil

http://www.inf.pucrs.br

Proposal of an Exploration of Asynchronous Circuits Templates and their Applications

Matheus Trevisan Moreira and Ney Laert Vilar Calazans

TECHNICAL REPORT SERIES

Number 077

March, 2014

ABSTRACT

Asynchronous circuits are gaining relevance in the research community due to their ability to cope with technology-related problems that synchronous circuits fail to. In particular, two common asynchronous design styles have been explored in the past. The first, called quasi-delay-insensitive design, relies on dual-rail or similar scheme that encodes when the computation completes into the data representation itself using delay-insensitive codes. This design style can yield high-performance circuits, but they are much larger than their synchronous counterpart is and have very high switching activity. The second design style, called bundled-data relies on delay lines matched to individual clouds of combinational single-rail logic. The advantage of this approach is reduced area and activity switching. However, because it relies on a complex set of timing assumptions it faces technology-related problems similar to synchronous design. This document proposes a research plan to explore the design space of available asynchronous circuits' templates and, once the tradeoffs are evaluated, understand their applicability for coping with contemporary technologies challenges. It also includes the proposition of automated environments for the synthesis of asynchronous circuits targeting different templates.

1. Introduction

1.1. Motivation

The ever-increasing demand for more complex systems and the possibility of integrating billions of transistors in a single chip brought us to the boundaries of the synchronous paradigm capabilities. The efficient distribution of a global *clock* signal in a contemporary complex design can be a daunting task. Albeit techniques and tools to help this exist, they can lead to overheads in power and area [MAR06]. According to Amde et al. [AMD05] circuitry required for the correct distribution of the clock signal can represent up to 50% of the total power of a contemporary chip. As power budgets get tighter, motivated by battery-based applications demand, and performance gets over constrained by aggressive process variations, traditional design techniques prove to be unsustainable [EKE10] [CHA13]. In fact, the International Technology Roadmap for Semiconductors predicts that a shift on integrated circuits (ICs) design paradigm is required in order to provide further improvements [ITR11]. In this scenario, asynchronous techniques emerge as a promising solution to cope with technological problems faced by synchronous designers [BEE07], [BAI08], [JUN10], [LIA10], [CRO10], [MIN11], [SCH11], [JIA11], [TSU12] and [RAB13].

However, differently from synchronous circuits, asynchronous circuits can be implemented using a wide variety of templates. Two of these are of particular interest to the asynchronous research community and have been extensively explored in the past. The first, called quasi-delay-insensitive (QDI) design, relies on dual-rail or similar scheme that encodes when the computation is complete into the data representation itself using delay-insensitive (DI) codes. This design style is usually associated to high robustness, as it relies on relaxed timing constraints [MAR06], but they are much larger than their synchronous counterpart (often 4x larger) and have very high switching activity (see e.g. [STE09a] and [BEE10]). The second design style, called bundled-data (BD), relies on delay lines matched to individual blocks of combinational single-rail logic. The advantage of this approach is that the switching activity within the logic blocks is essentially the same as in synchronous design and control signals are decoupled from data, to enable a considerable reduction in the design complexity of control circuits. The Achilles heel of BD design, however, is that the delay lines must be conservatively implemented to have a delay longer than that of its controlled logic under all possible process, temperature, and voltage (PVT) corners. Consequently, in the presence of aggregated on-chip variations in contemporary technologies, delay lines must be implemented with large margins, potentially taking away much of the advantages of asynchronous design [MAR06].

Currently, a wide variety of templates is available in the literature for implementing QDI and BD asynchronous circuits, each with its own advantages and drawbacks. That means that the choice for an asynchronous template is not an easy decision. In addition, head-to-head comparison between different asynchronous templates, and with synchronous designs, is very scarce in current literature. In this way, understanding the tradeoffs between state-of-the-art asynchronous templates and the synchronous template can enable to identify the suitability of each template for different application requirements, such as low-power, high-speed, high-density and high-robustness. In this way, designers can more easily select the correct template for a given problem.

1.2. Objectives

The objective of this research is to explore the design space of state-of-the-art techniques for asynchronous design and, once the tradeoffs are evaluated, assess the suitability of different asynchronous templates for achieving different design constraints, such as high-speed, low power, high density and high robustness. In addition, an automated environment composed by electronic design automation (EDA) tools will be developed for supporting the evaluated templates. To evaluate the distinct templates, a test chip will be fabricated. Hence, the following strategic goals are defined:

1. Select a set of asynchronous circuit's templates from the state-of-the-art.
2. Provide an automated environment, including tools and libraries, for designing circuits that employ the selected templates.
3. Design and prototype a set of case study circuits, which target the selected templates, together with equivalent synchronous versions.
4. Compare the case studies in terms of suitability for high-speed, low power, high-robustness and high-density. Comparisons will use both simulation and measurements in fabricated circuits.

To accomplish these strategic goals, the following specific objectives should be fulfilled:

1. Explore state-of-the-art ICs design-flows and tools.
2. Prototype a synchronous test chip.
3. Explore and select a set of state-of-the-art asynchronous circuit’s templates.
4. Improve automation degrees for asynchronous standard cells design.
5. Have a set of components required by the selected templates available at the standard cell level.
6. Explore state-of-the-art techniques for low power and high-robustness design.
7. Propose a design flow for automating the task of synthesizing circuits using the selected templates.
8. Select and design a set of case study circuits for comparing the selected templates.
9. Design the synchronous version of the same case studies.

10. Send a test chip for fabrication.
11. Evaluate and compare the case studies through simulation and measurements on the fabricated test chip.

1.3. Structure of this Document

The remaining of this document is organized as follows: Section 2 presents basic concepts to ease the reading of this proposal. Next, Section 3 discusses related work and provides an overview of where the proposed work will contribute to the state-of-the-art. Section 4 presents the work that was already conducted in the context of the proposed work and Section 5 provides an overview of ongoing and remaining work. In this same Chapter, there is a proposal of schedule of activities for concluding the proposed work. Finally, Section 6 draws some conclusions and puts the proposed work in perspective with the state-of-the-art.

2. Basic Concepts

2.1. Synchronous Design

The synchronous paradigm is very attractive for designing pipelined digital circuits mostly due to its simplicity. In synchronous circuits, a global signal called *clock* controls the sequencing of events of all sequential components, which are usually edge triggered registers [RAB03]. This signal oscillates at a fixed frequency and duty cycle, both defined at design-time, which convey to the designer a discrete notion of time. In fact, this is the beauty in synchronous design. Since all registers switch at the same instant of time thanks to the *clock* signal, the delay of wires and combinational gates in a path between a pair of registers can be ignored, given that a set of timing constraints related to the *clock* signal are respected. For instance, assume the example circuit of Figure 1. In this circuit, the minimum and maximum delays of the combinational logic, *i.e.* the minimum and maximum delays from *Q0* to *D1*, are t_{CLmin} and t_{CLmax}, respectively. The delays t_{R0min} and t_{R0max} are the minimum and maximum propagation delays of register *R0*, *i.e.* the minimum and maximum delays from *D0* to *Q0*. Assume also that the setup and hold time constraints of register *R1* are $t_{R1setup}$ and t_{R1hold}, respectively, and that the clock *CLK* period is t_{CLK}. According to Rabaey *et al.* [RAB03], under ideal conditions where the *clock* edges occur in both registers at the same time, *i.e.* *CLK0=CLK1*, the correct functionality of such a circuit is guaranteed provided that

$$t_{CLK} - t_{R0\max} - t_{CL\max} \geq t_{R1setup} \quad (1)$$

and

$$t_{R0\min} + t_{CL\min} \geq t_{R1hold} \quad \mathbf{(2)}$$

In other words, (1) and (2) ensure that, respectively, the setup and the hold constraints of registers are respected.

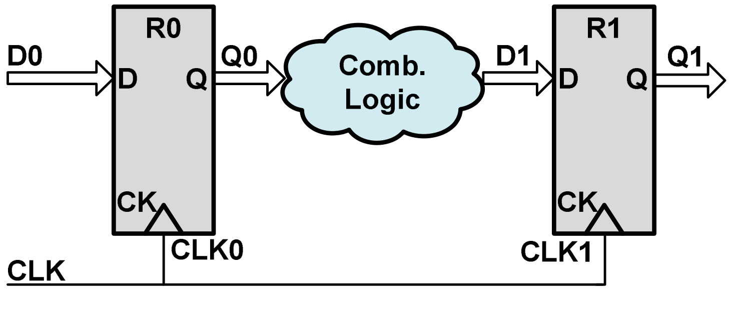


Figure 1 – Example of a synchronous datapath.

Unfortunately, clocks are never ideal. In fact, in a real circuit, transitions of *CLK0* and *CLK1* would usually lag in different proportions in relation to the reference *clock* signal (*CLK*). This is due to spatial variations in the arrival time of the clock signals, caused by static mismatches in the *clocks'* paths and differences in their loads. This phenomenon is classically called *clock skew* and is constant from cycle to cycle [RAB03]. In other words, if the active edge (the edge that triggers the register) of *CLK1* is delayed by a real number δ, then on the next cycle, it will be delayed by the same amount. Note that *clock skew* modifies the analysis of the example circuit of Figure 1 because registers *R0* and *R1* are no longer activated at the same instant of time. This requires adjustments in (1) and (2) and, as discussed in detail in [RAB03]. To guarantee the correct operation of the circuit, one must take into account the value of δ such that

$$t_{CLK} - t_{R0\max} - t_{CL\max} + \delta \geq t_{R1setup} \qquad (3)$$

and

$$t_{R0\min} + t_{CL\min} - \delta \geq t_{R1hold} \qquad \textbf{(4)}$$

Equation (3) indicates that bigger δ values have the potential to improve the performance of the circuit, since bigger maximum propagation delays t_{R0max} and t_{CLmax} can be tolerated for a same clock period t_{CLK}. In fact, this is correct, but increasing the skew has the side effect of making it more difficult to meet hold constraints. As (4) shows, the bigger the value of δ, the bigger the delays required for the combinational logic path to meet the equation. Also, δ can be a negative value, which relaxes hold constraints but tightens setup constraints.

Dealing with those problems was classically a low price to pay for the advantages of the synchronous paradigm. However, as reported by several authors (check [STE01], [STE09a] and [CHA10]), this price is considerably larger for modern technologies and the correct and efficient distribution of a global *clock* signal is getting prohibitively expensive. In fact, according to the International Technology Roadmap for Semiconductors (ITRS) in its 2011 edition [ITR11], a shift in integrated circuits (ICs) design paradigm seems to be inevitable. This is why asynchronous design techniques are receiving increasing attention of the very large scale integration (VLSI) research community.

2.2. Asynchronous Design

A digital circuit is *asynchronous* when no global or regionally global *clock* signal controls any sequencing of events. Instead, asynchronous modules use handshaking between its components to synchronize, communicate and operate [MYE01] [SPA01] [BEE10]. This means that each pair of registers communicates by explicitly signaling sending and receiving data. In "synchronous terms", the resulting behavior correspond to registers clocked only when and where needed. Such characteristic presents several advantages over the use of a global clock signal in modern technologies, as discussed in [SPA01], [MYE01], [STE01], [MAR06], [BOU07], [BEE10], [CHA10], [NOW11] and [CHA13]. However, differently from synchronous circuits (which basic assumption is the existence of a global *clock* signal), to implement an asynchronous circuit, several different templates are available.

The most basic and intuitive manner of implementing asynchronous communication consists in using two control signals in opposite directions, *request* (req) and *acknowledge* (ack). As Figure 2 shows, one protocol consists in an active element sending a request to synchronize with a passive element, which issues an acknowledgement when it is ready to communicate. Accordingly, communication can be based on either a 2-phase or a 4-phase protocols. The former is also known as transition signaling and the latter as level signaling [SPA01]. For instance, Figure 3 (a) shows an example of a 4-phase handshake communication. In this example, the active element starts with a requisition to communicate, rising the *req* signal. The passive element reads, processes the request and asserts the *ack* signal. When the active element reads the acknowledgement, it sets the *req* signal to low, which the passive element acknowledges by lowering the *ack* signal as well. After that, a new communication can take place at any moment. The transition signaling protocol is very similar to level signaling but its transitions reduced to half of the former. In this protocol, as Figure 3 (b) shows, the active element starts with a request, switching the logic value of the *req* signal. When the passive element reads the request, it sends an acknowledgement by switching the logic value of the *ack* signal. If data is to be transferred using handshaking, two approaches are usually employed: BD and QDI design.

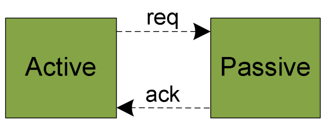


Figure 2 – Example of pure control communication through handshaking

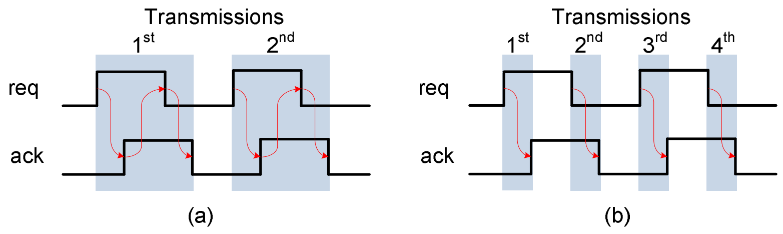


Figure 3 – Operation of the 4-phase (a) and 2-phase (b) handshake protocols.

2.2.1. Bundled-Data Asynchronous Design

In BD design, there are two possible configurations for transferring data: using push or pull channels. The former consists of two handshaking elements that transfer data, where data flows from the active to the passive. In the latter, on the contrary, data flows from the passive to the active element. Examples of push and pull data channels appear in Figure 3 (a) and Figure 3 (b), respectively. Note that in BD push channels, data in the inputs of the passive element must be valid before it receives a request. Similarly, in BD pull channels, data in the inputs of the active element must be valid before it receives an acknowledgement. This is typically done by employing delay elements for matching the delays of data and control wires.

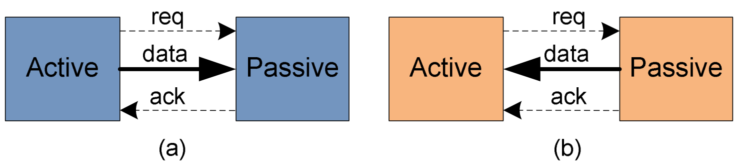


Figure 4 – Example of push BD data channel (a) and pull BD data channel (b).

As an instance, Figure 5 shows an example of a fragment of a BD circuit using a push channel. Accordingly, block *R0* signals requests to block *R1* for sending data through the *data* channel using the *req* wire. In such a scenario to guarantee that *R1* will correctly sample the transferred data, the *delay* line must wait longer than the critical path of the *Logic block*. This delay is typically adjusted using pairs of inverters or buffers. Figure 6 (a) shows an example of a 4-phase BD communication. First, information is inserted in the *data* channel. When the data is stable, the *req* wire is set to 1, signaling a request. This request is acknowledged by a transition to 1 in the *ack* signal. Next, the *req* signal must switch back to 0, which must be followed by the *ack* signal switching to 0. From this point on, a new communication can take place. Figure 6 (b), in turn, shows an example of a 2-phase BD communication. After data is stable in the *data* channel, a request is signaled by switching the logic value of *req*. When the acknowledgement is issued, i.e. after the *ack* signal has its logic value switched, a new communication can begin. Note that a similar analysis applies to pull channels.

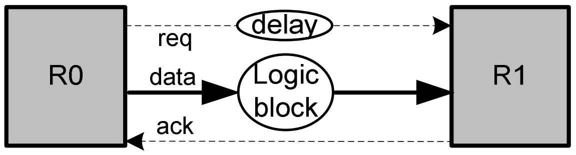


Figure 5 – Example of a bundled-data circuit fragment.

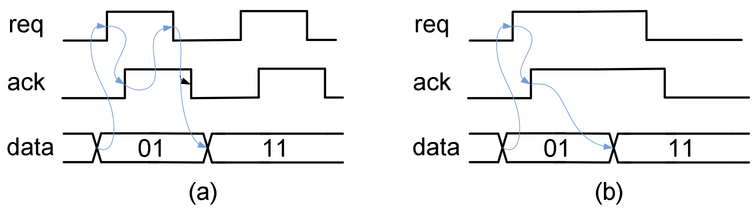


Figure 6 – Example of (a) 4-phase and (b) 2-phase bundled-data communications.

Starting from these assumptions, there are different ways to implement BD circuits, as described in [NOW11], each with its tradeoffs in terms of area, power and speed. The advantage of BD is that logic is designed similarly to what occurs in synchronous circuits, which easily adapting to use conventional tools for synthesizing and optimizing combinational logic. The drawback is that there is little support for correctly defining constraints for generating the delay line and, currently, its generation counts with small degrees of automation. A seminal work by Muller in [MUL57], proposed an asynchronous pipeline using C-elements in the control block. Another approach was proposed by Sutherland [SUT89], where special storage components are associated with a control block based on C-elements for implementing an asynchronous pipeline built with a BD style. The generated circuits are called *micropipelines*. Another recent alternative is the Mousetrap pipeline template [SIN07], which employs only conventional latches and XNOR gates for controlling local handshaking. Further details and circuit types are available in [NOW11].

2.2.2. Quasi-Delay-Insensitive Asynchronous Design

The drawback of the BD style is that the assumption of bundled control signals requires extra care with the computation of timing constraints between data and control signals and the implementation that respect these. Another possibility is the codification of the request/acknowledge signal within the data channel. This is the strategy adopted by QDI templates. Note that, similarly to BD, data can be transmitted through either push or pull channels. For the former, the request signal is encoded within the data, while for the later the encoded signal is the acknowledge. QDI templates require the choice for a handshaking protocol and a DI code to represent data. Different works proposed a variety of QDI templates in the past, as reported in [VER88], [BAI03] and [PON12]. However, 4-phase handshaking coupled to 1-of-n DI codes are the most usually employed scheme, because they allow reducing design complexity [MAR06]. Accordingly, the next discussion approaches only these choices.

In 1-of-n DI codes, data is represented using *n* wires and data validity is signaled by setting exactly one wire to 1, leading to codes with exactly *n* distinct values. This is equivalent to rising the request/acknowledge (for push/pull channels) signal in 4-phase BD circuits. In addition, absence of data is signaled by setting all wires to 0, equivalent to lowering the request/acknowledge signal in 4-phase BD. For instance, in a 4-phase 1-of-2 QDI template using push channels, the request signal is encoded into the data signals making use of two wires per data bit (*d.t* and *d.f* are the names of the wires). This encoding is also called *dual-rail*. As Table I shows, in this template, a logic 0 is represented by a low *d.t* and a high *d.f* while logical 1 uses an opposite encoding. A high transition of the request corresponds to *d.t* and *d.f* having different logical values. To signal a low transition of the request, named a *spacer*, d.t and d.f are set to logic 0 [SPA01]. Figure 7 shows an example of a 4-phase dual rail data transmission. Communication starts with all wires at 0, indicating absence of data (the spacer). Next, the sender puts valid data (in this case “01”) in the *data* channel. The receiver computes the request and acknowledges, setting the *ack* signal high. When the sender receives the acknowledgment, it transmits a spacer to finish the handshaking. The receiver computes the spacer and sets *ack* low. The sender can then start a new transmission. Note that because between each transmission all wires must be at 0, this protocol is also known as return-to-zero (RTZ).

Table I – 4-phase 1-of-2 RTZ encoding for one data bit.

Value	d.t	d.f
Spacer	0	0
0	0	1
1	1	0
Invalid	1	1

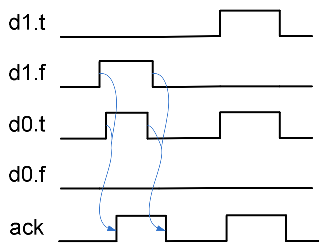
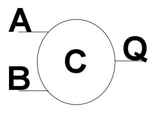


Figure 7 – Example of a 4-phase 1-of-2 (or dual-rail) RTZ QDI template communication for 2 data bits.

Because data signals in QDI circuits contain the encoded control signals, performing computation over such signals requires some care. This is different from BD circuits that can use logic gates similar to those employed in synchronous design. In fact, different design styles have been explored for designing combinational and sequential QDI circuits. Each of these is based in a specific hardware logic model of asynchronous template. Among the most employed templates, it is possible to cite: (i) the delay insensitive minterm synthesis (DIMS) [SPA01] [MYE01] [BEE10], (ii) the Null Convention Logic (NCL) [FAN96], (iii) the Weak Conditioned Half Buffer (WCHB) [MAR06] and (iv) the Precharged Half Buffer (PCHB) template [BEE10] [NOW11]. This Chapter ends with a brief overview for each of these relevant templates.

Delay-Insensitive Minterm Synthesis

The DIMS template is one of the most adopted for implementing QDI combinational logic. Some reasons behind its choice are its reduced design complexity and the fact that it requires only one specific asynchronous component, the C-element [BEE10], in addition to traditional components available in standard cell libraries, like gates AND, OR etc. This facilitates the adoption of conventional EDA tools and the use of semi-custom cell-based design flows. A C-element is a basic gate used for the synchronization of events in QDI design. Figure 8 (a) and (b) depict its symbol and truth table, respectively. The output Q of a 2-input (A and B) C-element only switches to 1 when both inputs are at 1, and to 0 when both inputs are at 0. Any other combination of inputs keeps the previous value of the output. C-elements are easily implementable at the cell level.



(a)

A	B	Q_i
0	0	0
0	1	Q_{i-1}
1	0	Q_{i-1}
1	1	1

(b)

Figure 8 – C-element (a) basic symbol and (b) truth table.

In DIMS blocks, all minterms of a logic function are first computed using C-elements. Next, the needed sum of the calculated minterms is computed using an OR gate similar to traditional two-level logic. This guarantees that the circuit will operate correctly and will respect the QDI template. In fact, DIMS can be used for any circuit that employs 1-of-n DI codes and 4-phase handshaking. For instance, Figure 9 (a) shows the DIMS implementation of a half adder circuit using a 1-of-2 4-phase template. In this example, the dual-rail sum output is 1 when exactly one of the inputs is 1, otherwise it is 0, and the carry output is 1 only when both inputs are 1, otherwise it is 0. This is computed by first producing all combinations of A and B lines (true and false wires *At*, *Bt* and *Af*, *Bf*), producing the minterms[1]. A 1 in the sum output ($S_{out}t$=1, $S_{out}f$=0) corresponds to exactly one input at 1 (*At*=1, *Bf*=1 or *Af*=1, *Bt*=1). In a similar manner, a 0 in the sum output ($S_{out}t$=0, $S_{out}f$=1) is generated when the inputs are the same, *i.e.* *At*=1, *Bt*=1 or *Af*=1, *Bf*=1. The remaining outputs are computed in a similar way. Note that the usage of a C-element for producing the minterms guarantees that a spacer will only be signaled in the outputs ($S_{out}t$=0 and $S_{out}f$=0 / $C_{out}t$=0, $C_{out}f$=0) when both inputs have a spacer (*At*=0, A*f*=0, B*t*=0, *Bf*=0). This is due to the basic functionality of this gate, as Figure 8 (b) shows, coupled to the functionality of OR gates, which will only signal a 0 when all their inputs are at 0.

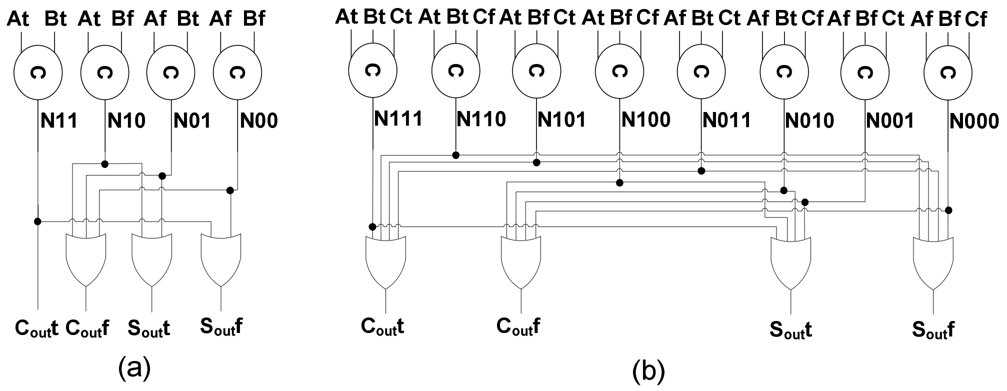


Figure 9 – Example 1-of-2 DIMS-based adders: (a) half adder; (b) full adder

A drawback of the DIMS model is the excessive use of C-elements for calculating all minterms of a given function. This usually leads to large delay, power and area overheads. Take for instance a more complex

[1] Remember that A lines or B lines are never 1 at the same time. Thus, no minterm is necessary for A_t-A_f or for B_t-B_f.

example, the full adder presented in Figure 9 (b). Since this function has 3 inputs, computing all minterms requires 2^3 3-input C-elements. Also, as the number of minterms increases, the number of inputs in the OR gates used to generate the sum of the minterms is also augmented. Depending on the function, this can lead to huge circuits for medium complexity functions. Moreover, these problems can get even worse when using more complex DI codes, where the number of minterms can become much larger.

Null Convention Logic

Another manner of implementing combinational logic for QDI circuits, called NCL, was proposed by Fant and Brandt [FAN96], and has been explored by different researchers, as reported in several works such as [LIG00], [KON02], [BAN07], [BAI08], [CHE08], [PAR12] and [REE12]. In fact, Theseus Logic was an enterprise that successfully prototyped many test chips based on NCL logic and had a synthesis flow for automating part of this process [LIG00]. When compared to DIMS, the NCL model enables the implementation of more efficient combinational QDI logic in terms of delay, power and area tradeoffs. A very important aspect of this model is that it relies on the usage of components at the cell level, also enabling cell-based design approaches.

In NCL design, basic components are sometimes called *threshold gates*, but this is imprecise. In fact, NCL gates couple a threshold logic function (TLF) [HUR69], with positive integer weights assigned to inputs, to the use of a hysteresis mechanism to guarantee a QDI compatible behavior. A TLF *t* is an *n*-variable unate function that implements a Boolean function defined by a threshold value *T* and a specific weight *w_i* assigned to each variable *x_i* such that:

$$t = \begin{cases} 1, & \sum_{i=1}^{n} w_i x_i \geq T \\ 0, & otherwise \end{cases} \qquad (5)$$

Figure 10 (a) shows the generic NCL gate symbol. Here, *N* is the number of gate inputs, *M* is either the gate threshold or a threshold function, and each input has weight *w_i*. Wherever no weight is specified, *w_i*=1 is assumed. Weights always come after the *w* specifier. The output switches to 0 when all *N* inputs are 0 and to 1 when the sum of weights for inputs at 1 reaches threshold *M*, or satisfies the threshold function. Otherwise, the previous output value is maintained (just like in C-elements). This enables QDI circuit design using 1-of-n data encoding and 4-phase handshaking. In this way, the ON-set of an NCL gate is defined by the ON-set of a TLF and the OFF-set consists of the minterm corresponding to all inputs at 0.

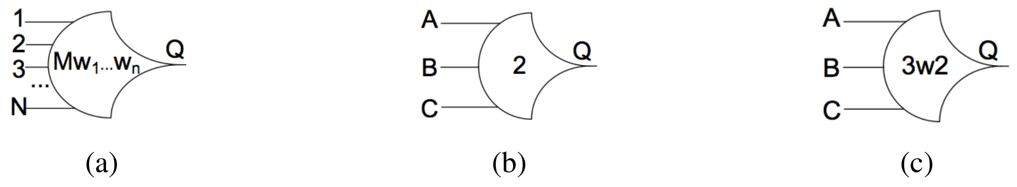


Figure 10 – Symbols for: (a) generic NCL gate symbol, (b) 2-of-3 NCL gate and (c) 3-of-3 with weights (A=2, B=1, C=1) NCL gate.

For instance, say that the threshold of a 3-input NCL gate is 2 and all inputs have weight 1. In such a gate, showed in Figure 10 (b), the output will only switch to 0 when all inputs are at 0 and to 1 when at least 2 of the inputs is at 1. Now, consider a 3-input (A, B, C) NCL gate with threshold 3, where the weight of the inputs is (2, 1, 1) respectively. In this gate, shown in Figure 10 (c), the output will only switch to 0 when all inputs are at 0. However to switch to 1, input A necessarily needs to be at logic 1, together with any of the remaining inputs, B or C, to reach the function threshold. Note that for such gate only weights bigger than 1 are made explicit in the symbol. NCL gates include OR gates as special cases (NCL gates with *M*=1 are in fact OR gates), as well as basic C-elements (NCL gates with *N*=*M* are in fact n-input C-elements). Besides, NCL gates can have negated inputs and outputs. For example, an NCL gate with *N*=1, *M*=1 and a negated output is in fact an inverter.

1-of-n 4-phase QDI combinational blocks can be built by combining NCL gates, based on their TLFs. For instance, a 1-of-2 4-phase 2-input AND can be build by using 2 NCL gates as Figure 11 (a) shows. Note that the true output *Qt* is computed using a 2-of-2 NCL gate which inputs are *At* and *Bt*, *i.e.* the output will only be 1 when both inputs are 1. The false output, in turn, employs a 3-of-4 gate with input weights (2, 2, 1, 1). Note that the false inputs *Af* and *Bf* are the ones connected to the first inputs of the NCL gates (those with weight=2), while the true inputs *At* and *Bt* have weight 1 in the gate. This means that the gate will only switch to 1 when both inputs have valid values and at least one of them is 0 (*Af*=1, *Bf=1* or *Af*=1, *Bt=1* or *At*=1, *Bf=1*). In addition, spacers will only be generated in the outputs when both inputs have spacers. This is guaranteed by the fact that the output of NCL gates will only switch to 0 when all inputs are at 0. A half adder can be implemented in a similar manner. As Figure 11 (b) shows, the AND block of Figure 11 (a) can be used for generating the carry signal ($C_{out}t$ and $C_{out}f$). The sum can be computed using two NCL gates that use an ANDOR function. Consider generating the true sum output, where $S_{out}t$ switches to 1. This occurs either when *At*=1 and *Bf=1* or when *Af*=1 and *Bt=1*. The false sum output ($S_{out}f$), on the other hand, switches to 1 when either *At*=1 and *Bt=1* or when *Af*=1 and *Bf=1*. Note that, in comparison to the DIMS approach, showed in Figure 9 (a), complexity to design a combinational block using NCL is considerably reduced and requires a smaller number of logic gates.

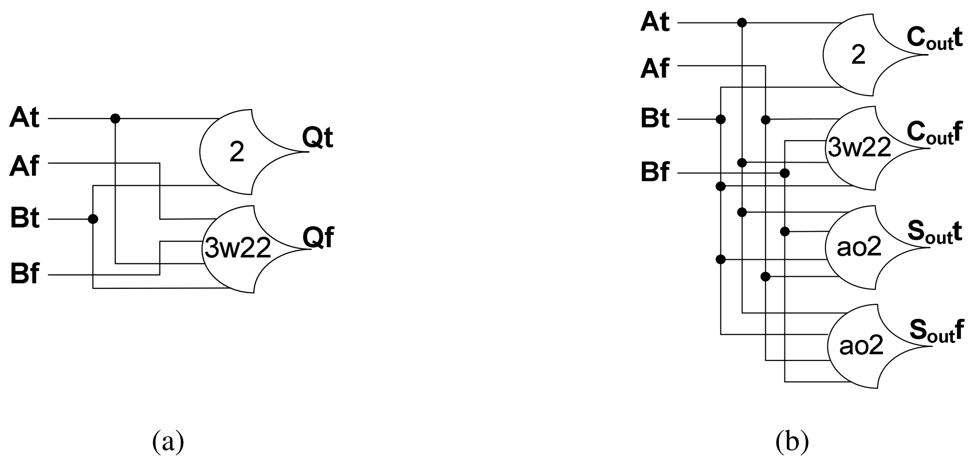


Figure 11 – Examples of 1-of-2 4-phase QDI circuits built with NCL gates: (a) 2-input AND gate; (b) half adder combinational block.

Weak-Conditioned Half-Buffer

While DIMS and NCL are employed for combinational logic, sequential QDI circuits are often designed using the WCHB template. Among other models, discussed in detail for example in [YAH06], this template is advantageous because it enables a reduced design complexity, due to the enforcement of a design methodology, similar to the one employed in conventional sequential circuits. Another advantage is the fact that WCHB counts today with an automated design flow associated to it that uses the Cadence Framework [THO12], which will be explored in Section 3. Similarly to DIMS, the WCHB template also requires only C-elements other than conventional logic gates for its implementation and is typically used together with both DIMS or NCL logic blocks. For example,

Figure 12 (a) shows the schematic of a 1-of-2 4-phase QDI 1-bit WCHB. The reset signal ensures that after reset the output will have a spacer. In such condition, the *INack* signal will have a 1, signaling to the previous WCHB that new data can be transmitted. Valid data will only be written on the output when it is available on the inputs (*INt* and *INf*) and the next WCHB can receive new data, i.e. *OUTack*=1. As soon as new data propagates to the output, the *INack* signal switches to 0, signaling to the previous WCHB that a spacer may be sent. Accordingly, spacers will only be written to the output of a WCHB when a spacer is available on the inputs and the next WCHB can receive a spacer (*OUTack*=0).

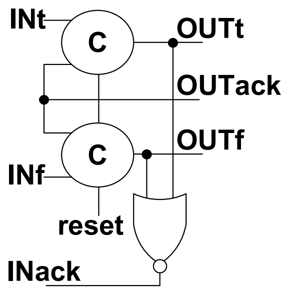


Figure 12 – Schematic of a 1-of-2 4-phase QDI 1-bit WCHB.

For composing WCHBs with the capacity of storing more than 1 bit of data, the *OUTack* control signal may be shared by all 1-bit WCHBs and their *INack* signals can be merged using C-elements, as discussed in [MAR06]. Note that two successively connected WCHBs always store a valid data and a spacer at any moment, or vice-versa. This is why this latch is a half-buffer. Another consequence is that a QDI pipeline with *n* stages based on WCHBs can contain at most n/2 valid data items.

Pre-Charged Half-Buffer with Dynamic CMOS Logic

The previous models are all based on static logic design. Albeit they allow a straightforward implementation, somewhat similar to what most synchronous designers are used to, they usually lead to large overheads in latency, given the need to compute data validity for each pair of registers at each stage. Another alternative is the use of the PCHB template with dynamic CMOS logic. In fact, several works demonstrate the advantages of this model to obtain high speed and low power circuits, as discussed in [BEE10]. Also, PCHB-based circuits have been extensively explored by Fulcrum (recently bought by Intel), which developed commercial product that employs QDI PCHB-based logic. In fact, there is a complete design flow for designing such circuits, as Section 3 discusses.

A PCHB-based circuit employs basic components implemented at the cell level, usually in the form of domino logic. These cells implement functions and assume the use of a 1-of-2 (or 1-of-4) 4-phase QDI template. As an instance, Figure 13 shows the example of a domino logic cell that implements a 2-input dual-rail OR gate. Upon reset, the enable (*en*) signal is low, generating a spacer on the outputs of the gate and signaling 0 in the validity output (*V*), which is used for enabling neighbor gates. This is the pre-charge phase, where the internal node, just before the output inverter, is set to 1. As soon as the enable signal goes high, signaling that there is valid data on the inputs, the cell evaluates the input through the NMOS network. For instance, if both inputs are at 0 (*Af*=1 and *Bf*=1), the internal node before *Qf* is discharged, switching *Qf* to 1, while the value of *Qt* is maintained by the memory scheme composed by the feedback loop with a weak feedback inverter. Note that as soon as valid data is signaled in the output, the validity output switches to 1. More information on implementations of control blocks for generating the enable signals can be found in [BEE10], [BEE11] and

[NOW11].

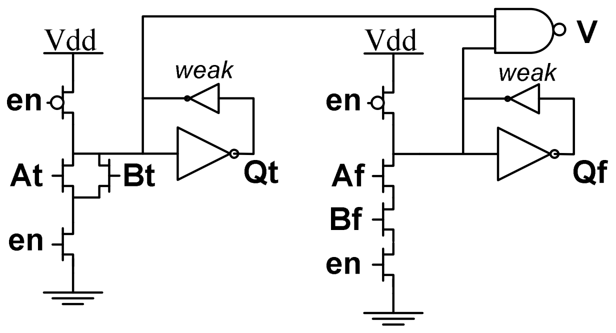


Figure 13 – Example of 1-of-2 4-phase PCHB domino logic cell implementing a 2-input OR gate [BEE11].

3. Related Work

This Section presents a set of works related to the tasks conducted in the context of the proposed work. Four main subjects were investigated for this purpose: asynchronous cell libraries and related automation tools, BD synthesis tools and design flows, QDI synthesis tools and flows and comparisons between asynchronous and synchronous circuits templates.

3.1. Asynchronous Cell Libraries

3.1.1. TIMA

The only fair way to widespread experiments in asynchronous application specific integrated circuits (ASICs) is to have available an asynchronous standard cell library. From this assumption, TIMA and LETI (both French laboratories) developed a library to support such designs. A 130nm gate length version of this library, called TAL-130 is presented in [MAU03]. In this work, the flow used to design the library is detailed and compares the results of implementing QDI circuits using TAL-130 and using a standard synchronous library (through AO222 gates).

At present, a 65nm transistors gate length version of this library has also been designed. This library is called TAL-65 and is fully validated through simulation and silicon implementations of QDI designs. Two cell sets compose the library, a set of C-elements and a set of latches. Several variations of C-elements are available, which are typically required for building control blocks. Among these are two-, three- and four-input cells, including settable and resettable gates. The drawback is that this library is not freely available. The information obtained about it was courtesy of TIMA and LETI. Also, there is no information available on how the cells were designed and/or of the availability of an automated design flow for the cell generation process.

3.1.2. CellTK

The work presented by Karmazin et al. in [KAR13] proposes the automated layout generator cellTK. The tool automatically implements the physical layout of asynchronous netlists. A drawback is that this generator is not compatible with semi-custom techniques and tools for asynchronous design automation proposed to date as it employs a non-standard flow, rather than a cell-based semi-custom flow. Also, the tool provides no support for automatic transistor dimensioning and there is no mention about how the generated layouts can be characterized to obtain power and timing models, which limits its usability to approaches close to full-custom. Another drawback is the area, energy and delay penalties imposed by cellTK, as reported by the authors, 51%, 12% and 9% in average, respectively, when compared to manual design.

3.1.3. USC Asynchronous Cell Libraries

The University of Southern California (USC) Asynchronous CAD and VLSI group has an extensive work on asynchronous design. It has reported the design of at least two cell libraries for asynchronous templates. The first library was designed in the context of the Ph.D. Thesis of Ferretti [FER04] and includes a set of basic cells for implementing circuits using a specific QDI template called single track full buffer (STFB). In his Thesis, Ferretti describes all techniques employed for transistor sizing and noise and performance analysis during the development of the cell library. For the sake of validation, a test chip was fabricated using the cells in a design with 260,000 transistors [FER06]. The library is freely available through MOSIS and targets the Taiwan Semiconductor Manufacturing Company (TSMC) 0.25 µm bulk CMOS process. A drawback is that there are no power and timing models available for the cells, as reported in [FER04]; only layout, schematic and symbol views are available. Also, the design of the library was completely handcrafted and adding cells or extending the library to other technologies can be very laborious. Additionally, another drawback is the lack of EDA tools for the STFB template.

A second library, which is of particular interest of for the development of the proposed work, is reported in [BEE11]. This work describes an automated design flow that targets PCHB design, and a library of basic components implemented at the cell level. The interest in such library comes from the fact that PCHB design is usually associated to high-speed circuits, providing feasible advantages for asynchronous design. Besides, cells of this library served to design test chips in Fulcrum and are currently employed in a commercial switch of Intel. The proprietary library was designed for the TSMC 65nm bulk CMOS process, targets 1-of-2 4-phase QDI and, according to Beerel et al. in [BEE11], implements all 2-bit and 3-bit logic functions. The library also includes control circuit□cells and C-elements with up to 4 inputs. Dedicated cells for implementing buffers are also included, together with specialized scan cells for enabling the testability of circuits built with the library. An advantage is that there is a tool for characterizing PCHB basic cells, as reported in [PRA07]. Drawbacks are that the library is proprietary and there is no reported support for automatic transistor dimensioning and cell layout generation. In this way, access to this library is very limited and extending it to other technologies can be very laborious.

3.2. Bundled-Data Synthesis

Currently, several design flows for asynchronous circuits are available, including EDA tools and synthesis methods. However, a drawback of BD approaches is that, albeit several works discuss their benefits over QDI approaches and propose different models for BD design [STE09a] [NOW11], there is very little automation associated to their construction. In fact, the majority of the BD approaches rely on manual methods for implementing the circuit. In addition, these works rarely explain the way they deal with timing constraints definition, which are the crucial point in BD design. In this way, the design of a BD circuit is usually associated to iterative, manual labor for defining and ensuring timing constraints.

3.2.1. Desynchronization

The desynchronization method proposed in [COR06], which uses as input typical synchronous circuit descriptions, is an example of automated flow for implementing BD circuits. After logic synthesis, the description is automatically modified, replacing flip-flops by latches and placing C-elements for implementing the local handshake control blocks. Additionally, combinational logic delays are matched to their respective synchronization lines. The resulting description can be synthesized using conventional tools. A drawback is that the resulting circuit usually presents low performance and there is little support to the definition of timing constraints. As mentioned before to ensure the respect of such timing constraints is a critical step in BD design wider adoption. Another drawback is that the method is restricted to the micropipeline BD template, which considerably limits design space exploration and popularity of the method.

3.2.2. Stevens et al. Asynchronous Design with Clocked CAD Flows

Given the necessity for a design flow for asynchronous circuits compatible with mature and consolidated CAD tools, Stevens et al. [STE09b] propose a methodology based on formal verification and relative timing to define and ensure the needed timing constraints to support BD design with traditional clocked CAD. An advantage of the proposed method is its generality regarding BD models, as it relies on formal specification and verification of timing constraints. In fact, the proposed method is useful to verify QDI designs as well. However, we focus on its ability to deal with BD designs, because these are implementable similarly to synchronous ones in a straightforward manner. There is a vertical compatibility of the proposed flow with BD templates, which does not apply to QDI ones. This is because the latter requires special care for synthesizing combinational and sequential blocks, as Section 2.2.2 explains. A drawback of the proposed approach is that there are no associated automation tools available to its application, requiring its manual application.

3.2.3. Asynchronous Constraints for Design Compiler

Due to the lack of tools for supporting BD design and providing some degrees of automation, the GAPH research group started to work on an environment with a set of scripts compatible with Synopsys Design Compiler for defining and verifying timing constraints. This environment is called Asynchronous Constraints for Design Compiler (ACDC) and was first discussed in [GIB13]. The ACDC approach is similar to what the current literature reports as manually done for BD designs. In fact, the flow is very similar to the one described in [STE09b] and is general, regarding different BD models, but it counts with automated scripts for ensuring that constraints are met. This is very advantageous as it reduces manual labor associated to BD design. Currently, a manually generated XML file specifies the set of timing constraints. ACDC then takes as input this XML and generates TCL scripts capable of checking, setting and reporting the defined constraints inside Synopsys tools. Gibiluka [GIB13] reported the use of ACDC in the design of a BD Network-on-Chip (NoC) router and this provided promising results in terms of design automation as well as in the quality of the resulting circuit. A drawback is that the specification of constraints in the XML file is still a manual step. However, improvements in ACDC are under way, for providing some degree of automation in timing constraints detection.

3.3. Quasi-Delay-Insensitive Synthesis

Many works on QDI design automation are available in contemporary literature. These works either propose dedicated sets of synthesis tools or design flows using conventional synthesis tools for implementing QDI circuits. Such tools and flows rely on one or on a combination of different asynchronous templates for QDI design. However, due to the lack of interest in asynchronous circuits in the 80s and 90s, these tools and flows are still in their early stages of development and are usually not mature enough for adoption in the design of large-scale commercial ICs, with the exception of the Proteus environment [BEE11]. This Section presents an overview of some of the most relevant available tools and flows. Note that this document disregards some current design flows such as the one commercialized by Tiempo [TIE14], due to the limited access to them.

3.3.1. Balsa

An example of a comprehensive framework for QDI design is Balsa [BAR98] [BAR00] [SPA01], proposed in the University of Manchester. It uses an extension of the Philips Tangram language [BER91]. Balsa includes a language for describing asynchronous hardware, also called Balsa. The language allows describing several classes of circuits in high levels of abstraction. The synthesis tool included in the Framework adopts a syntax-direct translation approach, which maps language constructions to handshake components [SPA01] [BEE10]. Such components employ the WCHB model for sequential blocks and DIMS for combinational ones. Syntax-direct translation is quite interesting, since modifications in the description lead to predictable modifications in the generated hardware, contrary to conventional hardware description languages, such as VHDL. Balsa allows synthesizing circuits for FPGAs and for CMOS technologies [BAR00]. For the latter, a specific set of components needs to be available at the standard cell level, for different DI codes and handshake protocol combinations [SPA01] [BEE10].

Unfortunately, it is the Author's own experience that circuits generated by Balsa usually impose very big area and power overheads. Moreover, the tool does not allow power and timing analysis, which prevents proper dimensioning of the cells that compose the generated circuit. Besides, conventional CAD systems fail to dimension the cells of such circuits because they cannot handle all the feedback loops that are present in the resulting asynchronous logic. In this way, the application of circuits generated with Balsa is limited.

Another option to synthesize Balsa descriptions is the open source back-end system Teak [BAR09]. It consists in a new target parameterizable component set and synthesis scheme that aims the improvement of circuits described in the Balsa language. The tool optimizes Balsa descriptions synthesis by replacing data-less activation channels with separate control channels. Albeit the Balsa framework supports different data encodings over 2-phase or 4-phase protocols, Teak implementations are typically QDI 4-phase dual rail asynchronous circuits. Hence, circuits synthesized through this tool are limited to that choice of protocol and data encoding. Another drawback is that in its current version Teak generates circuits with performance inferior to those generated by the Balsa System [BAR09].

3.3.2. Pseudo Synchronous

Another alternative for enabling QDI design is the flow proposed by Thonnart et al. [THO12], which employs conventional tools for synthesis and power/timing analysis. The flow starts with a structural VHDL description of a circuit using WCHB for sequential logic and DIMS for combinational logic. The flow treats WCHBs as flip-flops and a virtual clock signal exists for guiding the synthesis tool. One of the drawbacks is the manual labor required by the flow, as it mandates the modification of timing and power model files for tricking the synchronous design tools. Also, several iterations are needed during the synthesis to achieve the desired results. Besides, in some cases the generated circuits violate the constraints specified by the designer, which is a consequence of using a framework deemed for synchronous design while implementing asynchronous circuits. Another consequence of using a synchronous approach is the fact that the used tools always try to optimize the worst-case delay, which considerably limits design space exploration.

3.3.3. Proteus

Proteus is a design flow proposed by Beerel *et al.* in [BEE11] that targets the PCHB model for synthesizing 1-of-2, 4-phase QDI circuits. The flow relies on the usage of a proprietary cell library, as described in Section 3.1.3, and uses conventional synthesis tools. The input for the flow can be either register transfer level (RTL) legacy code or a CSP-based Fulcrum proprietary language. From the specification, the circuit is synthesized focusing only on the true wire of the 1-of-2 design, similar to a conventional single-rail synthesis, and is mapped into components of an image library. At this stage, the synthesis tools are capable of performing logic optimizations, which allows the flow to take advantage of mature algorithms employed by these tools. Next, the netlist is converted to 1-of-2 and the image cells are replaced by physical PCHB cells for the physical design. Note that in this process there is a series of optimizations that are performed in the asynchronous logic, as discussed in [BEE11], using a set of proprietary tools. Such optimizations allow improvements that conventional tools would not be able to do, because they focus on worst-case delays, which is not sufficient for asynchronous design. Next, place and route of the optimized design can occur, using conventional backend tools and the models of the PCHB library.

As discussed previously, commercial application already used Proteus to produce real-world circuits and results are encouraging. A drawback is that during synthesis, tools focus only on the true wire of the 1-of-2 design, which prevents that false and true wires share logic blocks. This may not seem significant for small paths, but for deep logic paths, sharing logic blocks between true and false wires can provide optimization opportunities. Another drawback of using Proteus is the fact that PCHB components are not available in conventional libraries and there is a laborious work associated to their implementation.

3.3.4. Null Convention Logic Design Flows

NCL comprises a basic set of components that classically counts with 27 gates [SMI96] [REE12]. These components include C-elements and some of the conventional logic gates as special cases, more specifically OR gates. Several works associate this model to high-speed [JUN10] [MIN11], low power [JOR10] [LIA10] [XUG10] and robust implementations. NCL also allows easy development of a semi-custom standard cell-based design flow. It is thus gaining relevance in the asynchronous research community. In fact, several works propose automated flows and tools for designing and optimizing QDI circuits based on NCL, e.g. [CHE08] [KON02] [LIG00] [PAR13] [REE12]. The drawback is that most of these flows and tools perform logic optimizations before technology mapping, which is precisely the first step in the synthesis of a circuit that allows optimizations with realistic cost parameters of the target technology. Besides, they all rely on synthesizing the circuit as if it was a single rail version, then replacing single rail logic gates by equivalent NCL templates (combination of NCL gates). This prevents exploration of optimizations enabled by logic sharing. The only work the author could find that somewhat allows post-mapping optimizations is the work of Cheljoo and Nowick, described in [CHE08]. However, this work provides only basic optimizations again based on predefined logic templates. Thus, it allows performing logic optimizations by replacing predefined sets of templates when these are present in the netlist.

3.4. Asynchronous Circuits Templates Comparison

There is a currently limited set of works that compare different asynchronous circuit templates. In fact, no proposal reports head-to-head comparison of different templates, as far as the author could verify. Additionally, there are no directions on what template is more beneficial for different application requirements. There is, however, a common agreement that QDI is typically more robust than BD, at the expense of increased area and power [MAR06] [STE09a]. It is a common belief that QDI is typically more suited for voltage-scaling applications, due to its more relaxed timing constraints, albeit there are some works about voltage scaling in BD designs. As to operating speed, the work with Proteus [BEE11] clearly demonstrates the potentials of QDI design for high-speed applications. That said, there is also space for BD designs to explore similar strategies, *i.e.* rely on the usage of dynamic logic for improving performance. Also, 2-phase design is typically cheaper for BD, which can lead to further improvements.

Finally, when it comes to comparison of designs based on asynchronous templates with synchronous designs, the availability of works is also scarce. Some works do compare asynchronous and synchronous designs – see e.g. [AUL93], [BER99], [SHE07] and [GEB10] – the comparison typically relies on a single (sometimes not optimized) asynchronous implementation. That is, there is no exploration of the asynchronous design space. In fact, the lack of such comparisons limits the VLSI research community understanding of the applicability of the different available asynchronous templates in the state of the art. In this way, performing head-to-head comparisons of a set of case studies implemented using state-of-the-art asynchronous templates and their synchronous versions can provide breaking ground knowledge on the applicability of the asynchronous paradigm for coping with contemporary VLSI problems.

4. Current Progress

In view of the current available literature, there is a clear need for a better understanding of the boundaries between synchronous, QDI and BD design and their applications. To provide a fair comparison, the first step is to provide an environment for design automation that supports such templates. However, doing so for asynchronous design is not as straightforward as for the synchronous paradigm. This is mainly due to four reasons: (i) asynchronous circuits can be implementing through a variety of templates, (ii) each template presents different synthesis challenges, (iii) each template can require a different set of basic components and (iv) the design of these components can be quite complex as there is little support to them. In this way, there is an implied big effort for setting up an environment for asynchronous circuits design. This Section provides an overview of the work that was already conducted towards the defined objectives of Section 1.2.

4.1. Explore state-of-the-art IC design flows and tools

To understand the process of designing an IC, the Cadence Framework was employed. The author explored logic and physical synthesis tools through the design of several case studies, including an RSA cryptographic core, the Hermes NoC router [MOR04] and a set of arithmetic circuits. Several topics were studied during this activity, from which standoff: timing analysis (including clock related issues of setup and hold), power analysis, analog and mixed signal simulation and physical verifications. Note that the author already had some experience with these tools from works conducted during his computer engineering and M.Sc. courses, as reported in [PON10a], [PON10b], [MOR11a] and [MOR11b]. Such studies were also supported by a course taken by the author, offered by the graduate program, about VLSI design. In this course, the author participated in the design of a new NoC router called Yet Another Hermes (YeAH!), where he explored all the steps of a synchronous design, from specification to backend. Unfortunately, the router was not validated on silicon yet, only by post-layout simulation.

4.2. Prototype a synchronous test chip

In order to master the usage of commercial IC design EDA frameworks, and apply the knowledge obtained in Activity 1, the author participated of the design of a chip containing an MBLite processor [KRA09], an implementation of the MicroBlaze architecture. Accordingly, the author was responsible for the backend of the design, performing all steps from logic synthesis to sign-off. The MOSIS service [MOS14] served to fabricate the chip with target at the 130nm IBM bulk CMOS technology using standard cells, memories and pad libraries from ARM [ARM14]. Currently, a test board is under development for testing it. In addition, a technical report describing the work conducted for implementing the test chip is currently being written.

4.3. Explore and select a set of state-of-the-art asynchronous circuits' templates

In 2006 the GAPH group started its research in asynchronous circuits. In this occasion, it was detected the need for some components that were not available in conventional standard cell libraries to enable a semi-custom approach to asynchronous design. However, the development of a standard cell library is a very laborious work. In this way, a basic design flow was developed for automating parts of the process of generating such a library. The flow was called Asynchronous Standard Cells Enabling *n* Designs (ASCEnD) and was firstly proposed in [MOR10] and [MOR11a]. ASCEnD employs professional tools from commercial frameworks from Cadence and Mentor for some steps. However, conventional tools do not support some parts of the design flow of a basic cell for asynchronous design. Thus, specially designed tools were developed to increase the cell design abstraction level and the design flow automation. Also, during the M.Sc. course of the author, the ASCEnD flow was optimized to support low-power cells design, as discussed in [MOR11b].

The availability of a cell library allowed exploiting methods for synthesizing asynchronous circuits. First, state-of-the-art QDI methods were studied and in a second phase, new methods were proposed. The first approach to design asynchronous circuits was based on structural constructions were components of the ASCEnD library were manually placed in a VHDL description as discussed in detail in [PON10a], [PON10b] and [MOR12a]. All these circuits used DIMS for combinational logic and WCHB for sequential logic. All of these employed C-elements available in the ASCEnD library. Because we had different implementations of this primitive, some initial studies also included the usage of different C-element transistor topologies for the design of high-speed and low-power QDI design, as discussed in [MOR12a]. This enabled defining the tradeoffs between each topology. In essence, after concluding these studies, the preferred topology was the Sutherland C-element for DIMS and WCHB design. Experiments with dynamic C-elements were also conducted. However, during such experiments, results indicated that the electrical behavior of this gate was more complex than previously believed, as reported in [MOR13h]. In this way, the usage of C-elements was restricted to static implementations.

During the study of synthesis methods, a new design style for QDI circuits was proposed. It is based in a different way of generating 1-of-n DI codes, called return-to-one (RTO), as presented in the work in [MOR12b]. In fact, the approach consists of representing spacers with all wires at 1, rather than all wires at 0 and data validity is signaled by setting one wire at 0. The work presents experimental results that indicate the suitability of RTO for reducing power in QDI circuits. The RTO protocol is similar to RTZ. The only difference is that wire values are inverted compared to RTZ. Table II shows conventions for a 1-of-2 code based on RTO. N wires at 1 (all-1s) represent spacers. A valid 1 is denoted by *d.t* at 0 and a valid 0 by *d.f* at 0. As Figure 14 shows, differently from RTZ, RTO data transmission starts after the all-1s value is in the data channel. Accordingly, in the example, communication must start with all wires at 1, indicating absence of data. Next, the sender puts a valid value "01" in the *data* channel by lowering *d1.f* and *d0.t*. The receiver computes the request from this value and acknowledges setting the *ack* signal low. When the sender reads the acknowledgment, it transmits a new spacer to finish the handshaking. The receiver computes the spacer and sets *ack* high. The sender can then start a new transmission. In this way, RTO-RTZ domain interfaces for a same code requires only n inverters. As a generalization, an RTO *D.x* wire logical value is translatable from RTZ by:

$$\forall x, 0 \le x \le n-1 : RTO(D.x) = \neg RTZ(D.x) \qquad (6)$$

Here, expressions RTO(*D.x*) and RTZ(*D.x*) correspond to wire logic values in the RTO and RTZ domains, respectively. In this way, according to Martin [MAR90], the conversion of data from one domain to another is DI. Also, albeit so far the author explored only 1-of-2 DI codes, RTO design techniques can be easily adjusted to any 1-of-n code.

Table II – 4-phase 1-of-2 RTO codification for 1 bit of data.

Value	d.t	d.f
Spacer	1	1
0	1	0
1	0	1
Invalid	0	0

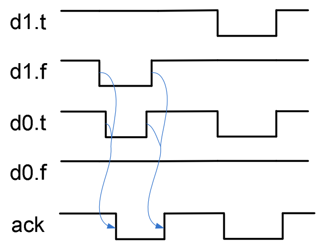


Figure 14 – Example of a 4-phase 1-of-2 RTO QDI template communication for 2 bits of data.

Another work by the author [MOR12c] evaluated the impact of using RTO in DIMS blocks. In this case, due to the inversion in the logic value of each wire, OR gates are replaced by AND gates. Note that we called the resulting logic model delay insensitive maxterm synthesis (DIMxS), as it relies on the generation of maxterms rather than minterms. The result of the function is then computed as a product of maxterms. For example, Figure 15 (a) and (b) show implementations of a 1-of-2 4-phase QDI half-adder using RTZ (DIMS) and RTO (DIMxS), respectively. Accordingly, the only difference in the implementation is the substitution of OR gates by AND gates. Using this approach, any RTO DIMxS logic block can be implemented. In fact, AND/NAND gates are classically preferred over OR and NOR gates in VLSI design. A stack of NMOS transistors is present in these gates, while ORs and NORs employ a stack of PMOS transistors. Because for contemporary CMOS technologies electron mobility is normally three times that of holes, NAND/AND gates are expected to present better power and delay tradeoffs than NOR/OR gates for the same silicon area. Therefore, RTO DIMxS circuits present a better power-delay tradeoff than equivalent RTZ circuits. In fact, as reported in

[MOR12c], the current required to propagate values and spacers for a DIMxS block is at least 44% and up to 48% lower than the one required by a similar DIMS block. Static power consumption was also beneficial for storing spacers, roughly 13%, and worse to store valid data, roughly 4%, when comparing RTO to RTZ. Also, a set of experiments recently conducted suggest that C-Elements in DIMxS logic blocks are over 300% more robust against transient faults when compared to those of DIMS blocks [MOR14d]. This demonstrates the potentials of employing RTO for constructing QDI circuits.

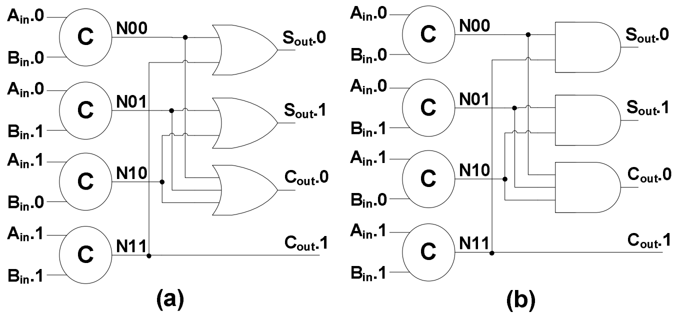


Figure 15 – Example of (a) DIMS and (b) DIMxS implementations of a 1-of-2 4-phase QDI half-adder.

Mixing RTO and RTZ in a same circuit has also been considered, as described in work recently approved for publication [MOR14a]. The obtained results suggest that albeit DIMxS provides better energy, leakage and speed tradeoffs than DIMS, WCHBs provide better tradeoffs for RTZ circuits. In this way, the work conducted so far indicates that using RTO for designing combinational logic blocks and RTZ for sequential logic blocks can provide the best tradeoffs. Also, because either C-elements, used in WCHBs, and AND/OR gates, used in DIMS/DIMxS, already have inverters in their outputs, the conversion between RTO and RTZ can come for free by using the node just before the output inverter of these gates. However, this will require a revision on transistor dimensions.

The author also explored the use of state-of-the-art EDA tools for asynchronous design. Accordingly, the ASCEnD library was integrated with Balsa [BAR98] [BAR00] and Teak [BAR09] EDA tools [MOR11a] [MOR11b]. Using these tools a network-on-chip router was described and synthesized targeting the STMicroelectronics 65nm bulk CMOS technology, as described in [MOR13e]. Note that the generated circuit relies on DIMS based combinational logic blocks and employs WCHBs for sequential logic. However, due to the low quality of the obtained netlist the author stopped using Balsa and Teak.

After experimenting with DIMS-based design, the author considered the NCL model, as reported in [MOR13a]. Simulation results of similar basic logic blocks implemented using both DIMS and NCL gates indicated that, as expected, the latter presented much better tradeoffs in terms of energy, leakage, speed and area. In fact, in all cases NCL proved to be superior to DIMS. Another style was proposed using RTO for constructing NCL logic, as discussed in [MOR13c]. The design style was called NCL+ and simulation results indicate that it allows solid reductions in static power when compared to classic NCL and provide better energy and speed tradeoffs. The NCL+ design style is similar to NCL, the only difference is the assumption of RTO rather than RTZ. In fact, NCL+ gates also have a threshold *M* (written inside each gate symbol). However, as defined in (6), the assumption of the RTO protocol mandates the switching function of an NCL+ gate to be the reverse of its NCL counterpart: the output will only switch to 1 when all inputs are at 1 and will only switch to 0 when threshold *M* is reached by the inputs at logic 0. For other combinations of inputs, the output keeps the previous value. The symbol that represents NCL+ gates appears in Figure 16. It is identical to that used for NCL, except for the "+" symbol on the top right corner.

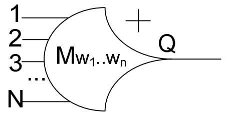


Figure 16 – Basic symbol of a NCL+ gate.

BD-based design styles have also been investigated. Three different BD styles have already been explored: the classic asynchronous pipeline proposed by Muller and Bartky [MUL57], the micropipelines proposed by Sutherland [SUT89] and the Mousetrap proposed by Singh and Nowick [SIN07]. Moreover, the author has participated in the design of a new intra-chip network router using a Mousetrap template to provide a better understanding the differences in challenges to design QDI and BD circuits. In this context, guidelines have been defined for specifying timing constraints during the synthesis of BD circuits and an automation tool has been proposed to enforce constraints closure, as explored in Section 3.2.3. Moreover, a deep discussion on this is available in [GIB13].

4.4. Improve automation degrees for asynchronous cells design

Since its initial version, the ASCEnD flow was optimized to support low-power cells design, as discussed in [MOR11b], and, later, for enabling the design of NCL gates [MOR13a] and NCL+ gates [MOR13c]. Besides, a new tool was recently added to the flow, due to problems faced during the electrical characterization of asynchronous standard cells using conventional tools. The tool is called Library Characterization Environment (LiChEn) [MOR12d] [MOR13d] and allows automatically extracting power, timing and capacitance figures from standard cells and exporting the generated models to the Liberty format, which is widely accepted by EDA vendors. The current structure of the flow appears in Figure 17. Note that this flow is similar to the one presented in [MOR13b], with the addition of the Astran tool for automatic layout generation. Astran [ZIE07], a tool for automatic layout generation designed in a partner research group from the Universidade Federal do Rio Grande do Sul was the latest addition to the ASCEnD flow and allows orders of magnitude time savings when compared to manual layout, as reported in [ZIE14a] and [ZIE14b]. Moreover, layouts generated by Astran proved to be more area, power and speed efficient than those manually designed. This is mainly due to two factors: (i) the manual layouts of ASCEnD were all designer by undergraduate students and (ii) Astran strives to minimize parasitic effects during layout generation.

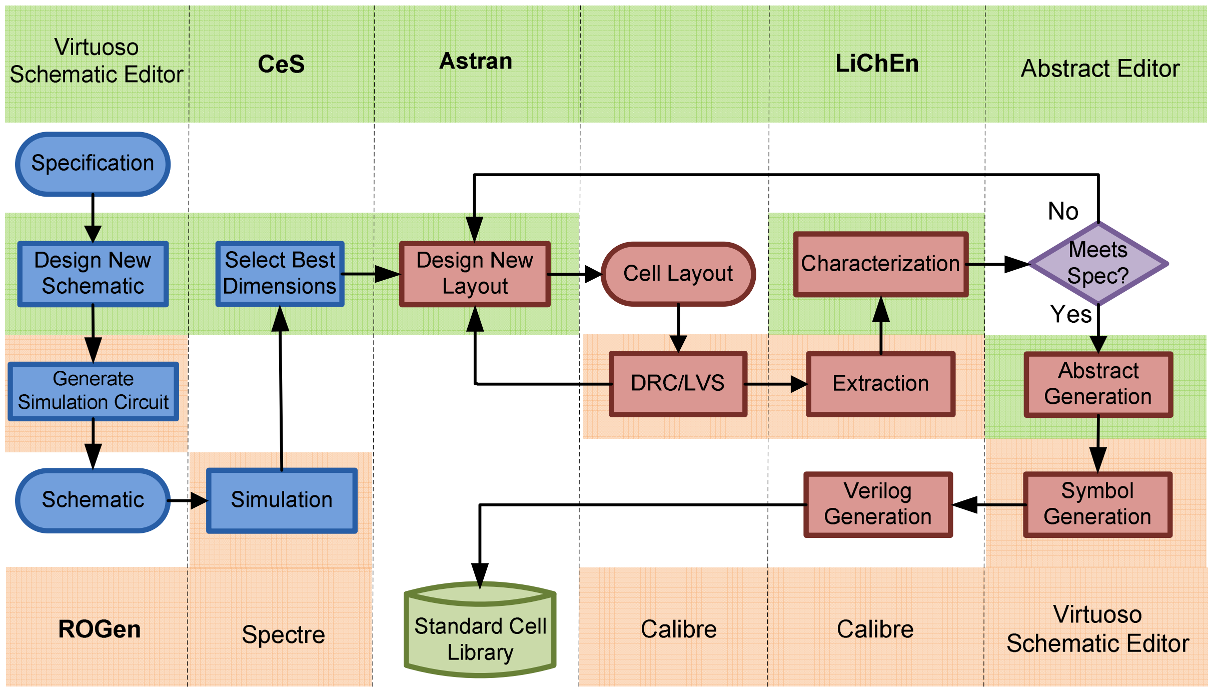


Figure 17 – General structure of the ASCEnD flow. Top and bottom rows indicate the tools used at each step. In-house tools have their names in boldface, while the others belong to a commercial framework, either from Cadence or from Mentor.

The design of a cell using the ASCEnD flow starts with logic and electrical specifications, which define the functionality and electrical requirements of the cell. That is, an initial schematic at the transistor level is designed using the Cadence Virtuoso Schematic Editor. This schematic, together with a set of technology specific configurations, is the input of an in-house tool called ROGen (from Ring Oscillators Generator), responsible for generating a test circuit and a set of measurements in SPICE. The generated description is simulated using Cadence Spectre and during simulation, power and delay reports are generated, according to the measurements specified by ROGen. These reports are the input of a second in-house tool called CeS (from Cell Specifier), which allows the cell designer to specify a cost function that can include delay and power metrics. Next, the tool automatically computes the best transistor dimensions for the cell, according to the provided cost function. Once transistors are dimensioned, the layout is automatically designed using Astran. The layout is verified using DRC and LVS tools to guarantee that no design rules are violated and that it implements the functionality specified in the schematic. Such verifications are currently performed using Calibre from Mentor. Once the layout is validated, RC parasitics are extracted using Calibre. The extracted circuit is employed for generating the standard cell delay and power models. This is automatically done by LiChEn (from Library Characterization Environment). The need for this tool was detected during the development of ASCEnD, when this step was done using conventional tools. The problem was that such tools did not recognize the functionality of standard cells for asynchronous circuits and ended up generating the need for a great deal of manual labor. After electrical characterization, physical models are generated to allow the use of cell by automated circuit Place and Route tools. This is done using Cadence Abstract Generator, which automatically produces Library Exchange Format files, widely accepted by commercial tools. Finally, a symbol is generated to ease the use of the cell in hierarchical circuit design. A behavioral Verilog is also created for enabling digital simulation. In this way, the current version of the ASCEnD flow allows the design of cells on demand, as the whole flow is automated and Verilog and symbol models can be reused.

4.5. Have a set of components required by the selected templates available at the cell level

Through the evolution of the ASCEnD flow happened the construction of a library (ASCEnD ST65) currently containing over 600 standard cells for the STMicroelectronics 65nm bulk CMOS technology, with the composition showed in Figure 18. ASCEnD ST65 has 504 different C-elements, 4 mutual exclusion components and 112 NCL gates (including NCL and NCL+ versions). It can support different BD and QDI styles and its contents evolved as the work described in Section 4.3 advanced, for supporting the research on different asynchronous design templates.

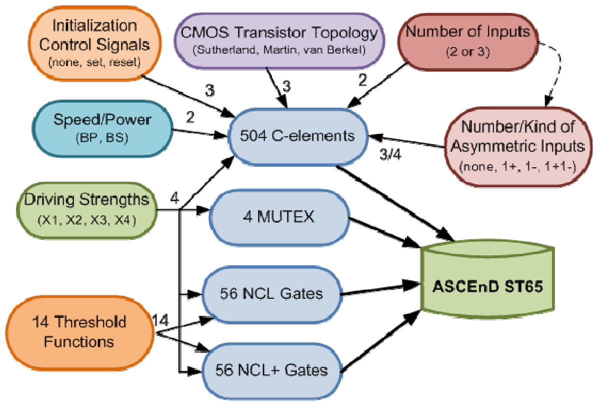


Figure 18 – Contents and structure of the current ASCEnD ST65 library.

The ASCEnD ST65 library was validated with the design of three different intra-chip network routers [PON10a] [PON10b] [MOR13e], two RSA cryptography circuits [MOR12a] and two tree adders. All circuits reached layout level and passed post-layout simulations. Aiming the design of a test chip that employs cells generated with the ASCEnD flow, a new library is under construction for the IBM 130nm technology, ASCEnD IBM130. The motivations for this is that one test chip per year can be fabricated free of charge for this technology through an institutional agreement with MOSIS [MOS14].

4.6. Explore state-of-the-art techniques for low-power and high-robustness design

An initial study towards low power QDI design enabled optimizing the ASCEnD flow to produce low power standard cells, as discussed in [MOR11b]. In fact, in its current version, the flow allows defining different cost functions during the design of standard cells, as discussed in [MOR13b]. A recent improvement of the flow enabled the incremental specification of cells in the CeS tool. This allows the designer to first filter dimensions targeting low power, balanced operation, etc. only then selecting the best dimensions among these for a second cost function.

The author has also worked on voltage-scaling for QDI design. As presented in [MOR13g], a first work evaluated the tradeoffs of different C-elements topologies under ingcreased supply voltages. This work enabled identifying that the Sutherland topology was more suited for low voltage operation and that the operating voltage representing the best tradeoff in terms of energy consumption, leakage power and delay was around 0.6 V (near-threshold). The minimum operational voltage was found to be 0.2 V for a 65nm selected technology.

Low power techniques for NCL design were also explored. A new topology for implementing NCL gates operating at low voltages has been proposed [MOR14b] and preliminary experimental results are promising. Differential NCL design was also considered, as there are works on current literature that demonstrate its suitability to low power design, such as [YAN10]. Accordingly, a new topology for implementing NCL gates and a new topology for enabling differential NCL design was proposed. Unpublished experimental results indicate that the topology allows further reductions of energy consumption and static power.

On the threads of robustness, ongoing work is the definition of a systematic for designing robust QDI circuits using RTO and RTZ. In this context, a methodology was defined for implementing robust NCL gates, for coping with charge sharing problems, as described in [MOR13f], and guidelines for employing dynamic C-elements have been defined, as explained in [MOR13h]. In addition, in the last months started the design of a new set of standard cells for integrating the ASCEnD library. This set comprises cells with Schmitt triggers in the outputs for enabling increasing tolerance to single event effects. Such choice was inspired in the work conducted by Kuang *et al.*, presented in [KUA07], where the authors propose the use of Schmitt Triggers in the output inverters of NCL gates for increasing their robustness against single event effects. Accordingly, their results suggest that, by doing so, the robustness of the gates increases by up to 100% in exchange of some area, power and delay penalties. However, the evaluation of these cells indicated that the experiments conducted in [KUA07] were not sufficient, as the authors did not include an analysis of single events on the output of the gates, only on their inputs. By considering those, we found that in such cases, a Schmitt Trigger on the output inverter of an NCL gate considerably reduces its robustness against single event effects (30% in average). This was done through the design of a set of case study gates and electrical simulations of a particle strike model, as described in [GAR08]. The obtained results also indicate that area, power and delay penalties are very substantial, up to 100% in most cases, and the efficient dimensioning of Schmitt Trigger transistors was a tricky task [GUA14]. In this way, the author concluded that the cost of employing Schmitt Triggers in

NCL gates is too high and improvements are not sufficiently worth it.

4.7. Propose a design flow for automating the task of synthesizing circuits using the selected templates

The lack of tools for the design of asynchronous circuits often requires the use of structural hierarchical description approaches, as those described in [PON10a], [PON10b] and [MOR12a], where standard cells of the ASCEnD library are manually instantiated using VHDL. Due to the fact that some components are often combined in certain higher level structures during asynchronous circuits design (e. g. flow control components, registers and combinational logic), a library containing macro blocks that implement some higher level functionalities was designed, to allow better reuse and modularity. The library, called LAMBDA, currently contains components like Merges, Forks, Joins, Arbiters, Multiplexers, Demultiplexers and Registers. Further information about the functionalities of these components exist in [SPA01]. All current LAMBDA components assume 4-phase 1-of-2 asynchronous template. However, due to recent advances in design techniques the development of LAMBDA was discontinued.

The latest contribution of the work conducted by the author is the development of a new design flow for automating NCL synthesis [MOR14c]. This design flow uses both NCL and NCL+ gates and counts with a set of automated scripts that allow synthesizing 1-of-2, 4-phase combinational blocks using conventional CAD tools. Some of the advantages of the flow are that it allows exploring logic optimizations of consolidated algorithms employed in such tools and sharing logic blocks between true and false wires automatically, which is not permitted by any other design flow available in contemporary literature. Figure 19 provides an overview of the proposed design flow. Note that it assumes the usage of the Cadence Framework in this discussion, but any other EDA Framework could be employed.

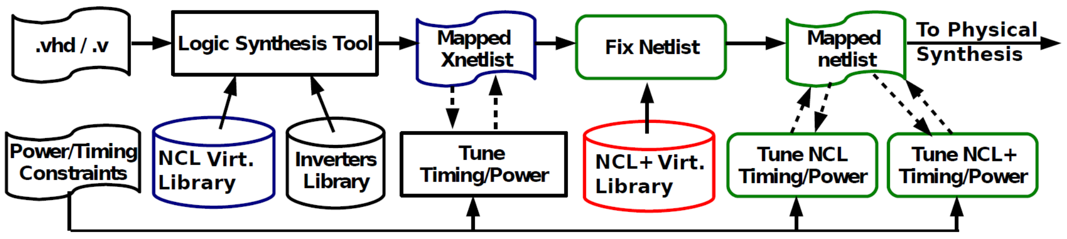


Figure 19 – Proposed design flow for NCL/NCL+ synthesis [MOR14c].

The input of the proposed design flow is an NCL Virtual Library together with a library containing inverters and buffers (as these gates do not jeopardize QDI properties) and a behavioral Verilog or VHDL description of an 1-of-n 4-phase QDI circuit. An NCL Virtual Library is a library of NCL gates mapped as conventional Boolean functions. Note that NCL/NCL+ threshold logic functions (TLFs) can be used to implement conventional Boolean functions, as described in [NEU13]. This allows the synthesis tool to employ conventional gates to generate a correct netlist.

Several case studies were synthesized and validated through simulation after processing with the above design flow. These case studies included an 8-bit ripple carry adder (8bRC), an 8-bit Kogge Stone adder (8bKS), a 32-bit Kogge Stone adder (32bKS), a 32-bit Arithmetic Logic Unit (32bALU) and a 16-bit shift and sum multiplier (16bMULT). The verified functionality of the design flow enables improving the quality of 1-of-n, 4-phase QDI designs based on the NCL model. For more details about this flow please refer to [MOR14c].

5. Ongoing and Remaining Work

To allow the exploration of the state-of-the-art and the definition of the focal point of this work, the activities conducted until this point, were mostly focused on extensive research, as discussed in Section 4. The progress of such work enabled the development of automated environments that help experimenting with diverse asynchronous circuits templates. Accordingly, it is important to highlight the following advances achieved in terms of asynchronous design support:

- Full automation of the ASCEnD flow: Cells can now be designed in a fully automated manner, allowing the generation of cells on demand. The development of LiChEn and the integration of Astran to the flow enabled this. The main point yet to develop to achieve a higher degree of generality is the support to multi-output cell design.
- Enrichment of ASCEnD ST65: The library counts with components that support many BD and QDI design templates.
- Proposition of an automated flow for NCL synthesis: The flow is compatible with tools from conventional EDA vendors and allows exploring novel aspects in NCL synthesis.

However, there is still a set of activities to conduct before the conclusion of this work. This Section proposes a schedule of activities for concluding this work

5.1. Explore and select a set of state-of-the-art asynchronous circuits' templates

From the explored asynchronous circuits' templates, the author selected 1-of-2 4-phase QDI based on NCL and PCHB as good candidates for the evaluation proposed in this work. This will allow exploring the tradeoffs of having static and dynamic QDI logic. The choice for the NCL template is due to the observed superiority of this model when compared to other static design approaches such as DIMS. In addition, the advances on QDI synthesis and the proposition of RTO and NCL+ enabled the development of an automated NCL design flow, which enables efficient design. As for PCHB, its successful adoption in commercial products, as reported in [BEE11], indicates the relevance of evaluating dynamic logic as well. Besides, experimenting with PCHB is supported by the Proteus flow, as presented in [BEE11]. Given that the author will follow a stage at USC, he will have the best opportunity to master the usage of this flow.

The author also perceives the need for defining one BD template, given that many works report the advantages of using BD design. However, the work conducted so far was not enough for precisely defining this choice. This definition will occur after the first half of the sandwich stage. Apart from the already explored templates, the author will also consider employing a new template that under development in a partnership between PUCRS and USC.

5.2. Have a set of components required by the selected templates available at the cell level

For the work with NCL design, most components are already available in the ASCEnD ST65. Any new gate that may be required is quickly implementable using the ASCEnD flow. For the PCHB design, on the other hand, there is no library support yet. Additionally, the ASCEnD flow is not compatible with PCHB design for two reasons. First, the flow does not support transistors dimensioning and electrical characterization of gates with more than one output and PCHB gates are typically multi-output. Second, Astran does not support PCHB layout, because these gates are typically too complex and require many transistors in their design. To arrange all transistors in an efficient manner, PCHBs usually have non-standard cell heights, which are not supported by the tool. Therefore, the design of PCHB gates will involve manual work. The advantage is that PCHB synthesis can rely upon a limited set of gates, which reduces the complexity of designing a library. Next, LiChEn is currently evolving to support multi-output gate characterization within the scope of an end of term work. The new version of the tool will be ready at the end of the first semester of 2014. For BD designs, once defining the template, at the end of the period of Activity 1, the required gates can be designed using the ASCEnD flow.

Another important ongoing work is the analysis of voltage scaling on NCL gates. In this context, the author is exploring techniques for designing gates that enable ultra-low voltage operation. Using these techniques, a whole library of NCL gates for ultra-low voltage operation will be devised at the GAPH group during an end of term work of a computer engineering undergraduate student. The library should be available in the end of the first semester of 2014. In this way, its usage on NCL design can also be evaluated in the context of the proposed work.

5.3. Propose a design flow for automating the task of synthesizing circuits using the selected templates

The NCL-based design will rely on the usage of the proposed design flow, described in Section 4.7 and in [MOR14c]. However, there are some aspects of this flow that need to be improved, such as the automated support to the design of sequential circuits. Increasing the degree of automation of this flow is an ongoing work expected to conclude by the end of the first semester of 2014. For PCHB design, the idea is to employ the Proteus flow. To do so, the author will first master the usage of the flow and then integrate the PCHB library designed in the context of Activity 2. Finally, the BD template will most likely employ ACDC [GIB13] for providing some degree of design automation. The expectation is that the environment for designing the circuits based on the chosen templates will be available by the end of the first semester of 2014.

5.4. Select and design a set of case study circuits for comparing the selected templates

The definition of a set of case study circuits for comparing the selected templates is a crucial task for the development of this work. Such definition can have a big impact on the quality of the obtained results. In other words, the selected case studies must allow a fair evaluation of the selected templates. Among the considered circuits are NoC routers, cryptographic cores and arithmetic blocks. There is a wide set of works dealing with the design of such circuits using asynchronous templates in contemporary literature. These circuits allow different degrees in terms of combinational logic complexity, sequential logic complexity and activity rate.

The case studies will allow explore different tradeoffs of the selected templates, enabling to scrutinize the advantages and drawbacks of each. The definition of the case studies will take place in the next few months such that in the end of the sandwich stage they are correctly implemented using the environment set up in Activities 2 and 3.

5.5. Design the synchronous version of the same case studies

The design and fabrication of the MBLite chip provided the author a better understanding of how IC design EDA frameworks work and how to correctly design synchronous circuits. In this context, the case study circuits defined in Activity 4 will also have a synchronous version designed, with the goal to allow synchronous-asynchronous comparisons. This will occur in parallel with the design of the asynchronous implementations. These should also complete by the end of the sandwich stage.

5.6. Send a test chip for fabrication

A test chip containing the asynchronous and the synchronous versions of the case studies will be designed and submitted to fabrication. This will be the concluding activity for the sandwich stage at USC. Note that, currently, it is assumed the availability of fabrication resources for the target technology.

5.7. Evaluate and compare case studies through simulation and measurements on fabricated test chip

The last activity before the conclusion of the proposed work will be the evaluation and comparison of the designed case studies. This will be done firstly through simulation and power, timing and area analyses of the designs. In this context, the circuits will be evaluated in terms of their silicon area, operating speed, dynamic and static power, robustness against PVT variations and radiation effects and suitability for low power operation. Currently, the author has available all the environments required to do such analyses. These settings count with integrated digital, analog and mixed-signal simulation environments that enable fast performance analysis using SystemC models and robustness analysis through Monte Carlo simulation and a particle strike model using SPICE. Besides, the GAPH group counts with a grid environment with more than 100 cores available, which will speed up the generation of simulation results. Next, evaluation of the designed test chip will take place in order to get more realistic parameters. The obtained results will allow an in-depth analysis of the tradeoffs of different asynchronous circuits' templates and the synchronous paradigm. Given that this activity will impose several challenges, a whole semester is dedicated only for that. This will be done after the sandwich stage.

6. Conclusion

The proposed work encompasses a set of original contribution to the research in asynchronous circuit design and design methods. Accordingly, upon conclusion of the work, the following items are expected to be available:

1. A fully automated design flow for implementing cells dedicated to asynchronous design supporting QDI and BD templates;
2. A new design flow for NCL-based circuits together with an open library of NCL and NCL+ gates;
3. An open access cell library of PCHB gates compatible with the Proteus flow and tool;
4. A better understanding of the tradeoffs for different asynchronous circuits' templates and their applicability;

Besides these, the following items can already be considered contributions of this work:

1. A fully automated design flow for implementing cells dedicated to asynchronous design with the limitation that cells must have only one output;
2. A new possibility for 4-phase 1-of-n QDI templates, called RTO, which allows better design space exploration for asynchronous circuits and enables better power and energy efficiency;
3. A new logic style for NCL design, called NCL+, that allows performing logic optimizations and enables better power, energy and delay compromises;
4. A new automated design flow for combinational NCL design that relies on the usage of conventional CAD tools;
5. A test chip completely designed by the GAPH research group.

7. References

[AMD05] M. Amde, T. Felicijan, A. Efthymiou, D. Edwards, L. Lavagno, “Asynchronous on-chip networks,” IEE Proceedings - Computers and Digital Techniques, 152(2), March 2005, pp. 273-283.

[ARM14] ARM. http://www.arm.com. 2014.

[AUL93] R. Auletta, B. Reese, C. Traver, “A comparison of synchronous and asynchronous FSMD designs,” in IEEE International Conference on Computer Design (ICCD), pp. 178–182, 1993.

[BAI03] W. Bainbridge, W. Toms, D. Edwards, S. Furber, “Delay-insensitive, point-to-point interconnect using m-of-n codes,” in 9th International Symposium on Asynchronous Circuits and Systems (ASYNC), pp. 132–140, 2003.

[BAI08] A. Bailey, D. Jia, S. Smith, H. Mantooth, “Ultra-low power delay-insensitive circuit design,” in 51st Midwest Symposium on Circuits and Systems (MWSCAS), 2008, pp. 503-506.

[BAN07] S. Bandapati, S. Smith, “Design and characterization of NULL convention arithmetic logic units,” Microelectronic Engineering, 84(2), Feb 2007, pp. 280–287.

[BAR98] A. Bardsley, “Balsa: An Asynchronous Circuit Synthesis System,” MSc Dissertation, Faculty of Science & Engineering, University of Manchester, 1998, 162p.

[BAR00] A. Bardsley, “Implementing Balsa Handshake Circuits,” PhD. Thesis, Faculty of Science & Engineering, University of Manchester, 2000, 187p.

[BAR09] A. Bardsley, L. Tarazona, D. Edwards, “Teak: A Token-Flow Implementation for the Balsa Language,” in International Conference on Application of Concurrency to System Design (ACSD), 2009, pp. 23-31.

[BEE07] P. Beerel, M. Roncken, “Low Power and Energy Efficient Asynchronous Design,” Journal of Low Power Electronics, 3(3), 2007, pp. 234-253.

[BEE10] P. Beerel, R. Ozdag, M. Ferretti, “A Designer's Guide to Asynchronous VLSI,” Cambridge University Press, 2010.

[BEE11] P. Beerel, G. Dimou, A. Lines, “Proteus: An ASIC Flow for GHz Asynchronous Designs,” IEEE Design & Test of Computers, 28(5), Sep 2011, pp. 36–51.

[BER91] K. van Berkel, J. Kessels, M. Roncken, R. Saejis, F. Schalij, “The VLSI-programming language Tangram and its translation into handshake circuits,” in European Conference on Design Automation (EDAC), 1991, pp. 384-389

[BER99] K. van Berkel, M. Josephs, S. Nowick, “Applications of asynchronous circuits,” Proceedings of the IEEE, 87(2), 1999, pp. 223-233.

[BOU07] F. Bouesse, N, Ninon, G. Sicard, M. Renaudin, A. Boyer, E. Sicard, “Asynchronous Logic Vs Synchronous Logic: Concrete Results on Electromagnetic Emissions and Conducted Susceptibility,” in 6th International Workshop on Electromagnetic Compatibility of Integrated Circuits (EMC), 2007, 5p.

[CHA10] I. Chang, S. Park, K. Roy, “Exploring asynchronous design techniques for process-tolerant and energy-efficient subthreshold operation,” IEEE Journal of Solid-State Circuits, 45(2), 2010, pp. 401–410.

[CHA13] K. Chang, J. Chang, “Synchronous-Logic and Asynchronous-Logic 8051 Microcontroller Cores for Realizing the Internet of Things: A Comparative Study on Dynamic Voltage Scaling and,” IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 3(1), 2013, pp. 23–34.

[CHE08] J. Cheoljoo, S. Nowick, “Technology mapping andcell merger for asynchronous threshold networks,” IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 27(4), 2008, pp. 659-672.

[COR06] J. Cortadella, A. Kondratyev, “Desynchronization: Synthesis of asynchronous circuits from synchronous specifications,” IEEE Transactions on Computer-Aided Design of Integrated Circuits, 25(10), Oct 2006, pp. 1904–1921.

[CRO10] J. Crop, S. Fairbanks, R. Pawlowski, P. Chiang, “150mV sub-threshold Asynchronous multiplier for low-power sensor applications,” in 2010 International Symposium on VLSI Design Automation and Test (VLSI-DAT), 2010, pp. 254-257.

[EDW06] D. Edwards, A. Bardsley, L. Janin, L. Plana, W. Toms, “Balsa: A Tutorial Guide,” 2006, 157p.

[EKE10] N. Ekekwe, “Power dissipation and interconnect noise challenges in nanometer CMOS technologies,” IEEE Potentials, 29(3), May 2010, pp. 26–31.

[ERN03] D. Ernst, K. Nam Sung, S. Das, S. Pant, R. Rao, P. Toan, et al., “Razor: a low-power pipeline based on circuit-level timing speculation,” in 36th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-36), 2003, pp. 7-18

[FAN96] K. Fant, S. Brandt, “NULL Convention Logic TM: a complete and consistent logic for asynchronous digital circuit synthesis,” in IEEE International Conference on Application-Specific Systems, Architectures, and Processors (ASAP), 1996, pp. 261–273.

[FER04] M. Ferretti, “Single-Track Asynchronous Pipeline Template,” PhD Thesis, Faculty of Electrical Engineering, University of Southern California, 2004, 170p.

[FER06] M. Ferretti, P. Beerel, “High Performance Asynchronous Design Using Single-Track Full-Buffer Standard Cells,” IEEE Journal of Solid-State Circuits, 41(6), June 2006, pp. 1444-1454.

[GAR08] R. Garg, S. Khatri, “A Novel, Highly SEU Tolerant Digital Circuit Design Approach,” in IEEE International Conference on Computer Design (ICCD), 2008, pp. 14-20.

[GEB10] D. Gebhardt, J. You, K. Stevens, “Comparing Energy and Latency of Asynchronous and Synchronous NoCs for Embedded SoCs,” in 4th ACM/IEEE International Symposium on Networks-on-Chip (NOCS), 2010, pp. 115–122.

[GIB13] M. Gibiluka, “Design and Implementation of an Asynchronous NoC Router Using a Transition-Signaling Bundled-Data Protocol,” Computer Engineering End of Term Work, FACIN, PUCRS, 2013, 110p. Available at http://www.inf.pucrs.br/~calazans/publications/2013_TCC_MatheusGibiluka.pdf.

[GUA14] R. Guazzelli, G. Heck, M. Moreira, N. Calazans, “Hardening NCL Gates Against SEEs Using Schmitt Trigger on Output Inverters: is it Sufficient?,” in 15th IEEE Latin-American Test Workshop (LATW'14), Fortaleza, 2014. Accepted for publication.

[HUR69] S. Hurst, “An introduction to threshold logic: A survey of present theory and practice,” Radio and Electronic Engineer, 37(6), June, 1969, pp. 339–351.

[ITR11] International Technology Roadmap for Semiconductors, “2011 Edition - Design Chapter,” available at http://www.itrs.net, 2013, 52p.

[JIA11] C. Jiaoyan, D. Vasudevan, E. Popovici, M. Schellekens, “Design of a Low Power, Sub-Threshold, Asynchronous Arithmetic Logic Unit Using a Bidirectional Adder,” in 2011 14th Euromicro Conference on Digital System Design (DSD), 2011, pp. 301-308.

[JOR10] R. Jorgenson, L. Sorensen, D. Leet, M. Hagedorn, D. Lamb, T. Friddell W. Snapp, “Ultralow-Power Operation in Subthreshold Regimes Applying Clockless Logic,” Proceedings of the IEEE, 98(2), 2010, pp. 299-314.

[JUN10] C. Junchao, C. Kwen-Siong, G. Bah-Hwee, J. Chang, “An ultra-low power asynchronous quasi-delay-insensitive (QDI) sub-threshold memory with bit-interleaving and completion detection,” in 2010 8th IEEE International NEWCAS Conference (NEWCAS), 2010, pp. 117-120.

[KAR13] R. Karmazin, C. Otero, R. Manohar, “cellTK: Automated Layout for Asynchronous Circuits with Nonstandard Cells,” in IEEE 19th International Symposium on Asynchronous Circuits and Systems (ASYNC), May 2013, pp. 58–66.

[KON02] A. Kondratyev, K. Lwin, “Design of asynchronouscircuits using synchronous cad tools,” IEEE DesignTest of Computers, 19(4), 2002, pp. 107-117.

[KRA09] T. Kranenburg, “Design of a Portable and Customizable Microprocessor for Rapid System Prototyping,” Master Thesis, Delft University of Technology, 2009.

[KUA07] W. Kuang, C. Ibarra, P. Zhao, “Soft Error Hardening for Asynchronous Circuits,” in 22nd IEEE International Symposium on Defect and Fault-Tolerance in VLSI Systems (DFT), pp. 273–281, Sep. 2007.

[LIA10] Z. Liang, S. Smith, D. Jia, “Bit-Wise MTNCL: An ultra-low power bit-wise pipelined asynchronous circuit design methodology,” in 2010 53rd IEEE International Midwest Symposium on Circuits and Systems (MWSCAS), 2010, pp. 217-220.

[LIG00] M. Ligthart, K. Fant, R. Smith, A. Taubin, A. Kondratyev, “Asynchronous design usingcommercial HDL synthesis tools,” in InternationalSymposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC), 2000, pp. 114-125.

[LOD12] F. Lodhi, O. Hasan, S. Hasan, F. Awwad, “Modified null convention logic pipeline to detect soft errors in both null and data phases,” in IEEE International Midwest Symposium on Circuits and Systems (MWCAS), 2012, pp. 402-405.

[MAR90] A. Martin, “The limitations to delay-insensitivity in asynchronous circuits,” in 6th MIT conference on Advanced research in VLSI, 1990, pp. 263-278.

[MAR06] A. Martin, M. Nyström, “Asynchronous techniques for system-on-chip design,” Proceedings of the IEEE, 94(6), 2006, pp.1089-1120.

[MAU03] P. Maurine, J. Rigaud, F. Bouesse, G. Sicard, M. Renaudin, “Static Implementation of QDI Asynchronous Primitives,” in International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS), 2003, pp. 181-191.

[MIN11] S. Mingoo, G. Chen, S. Hanson, M. Wieckowski, D. Blaauw, D. Sylvester, “CAS-FEST 2010: Mitigating Variability in Near-Threshold Computing,” IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 1(1), 2011, pp. 42-49.

[MOR04] F. Moraes, N. Calazans, A. Mello, L. Möller, L. Ost, “HERMES: an infrastructure for low area overhead packet-switching networks on chip,” Integration, the VLSI Journal, 38(1), Oct. 2004, pp. 69–93.

[MOR10] M. Moreira, “Design and Implementation of a Standard Cell Library for Building Asynchronous ASICs,” Trabalho de Conclusão de Curso. Engenharia de Computação - PUCRS. Dez 2010. 139 p.

[MOR11a] M. Moreira, B. Oliveira, J. Pontes, N. Calazans, "A 65nm Standard Cell Set and Flow Dedicated to Automated Asynchronous Circuits Design," in 24th IEEE International SoC Conference (SoCC), 2011, pp. 99-104.

[MOR11b] M. Moreira, B. Oliveira, J. Pontes, F. G. Moraes, N. Calazans, "Adapting a C-element Design Flow for Low Power," in IEEE International Conference on Electronics, Circuits, and Systems, Beirut (ICECS), 2011, pp. 45-48.

[MOR12a] M. Moreira, B. Oliveira, F. Moraes, N. Calazans, "Impact of C-elements in Asynchronous Circuits," in International Symposium on Quality Electronic Design (ISQED), 2012, pp. 438-444.

[MOR12b] M. Moreira, R. Guazzelli, N. Calazans, "Return-to-One Protocol for Reducing Static Power in QDI Circuits Employing m-of-n Codes," in 25th Symposium on Integrated Circuit and Systems Design (SBCCI), 2012. 6p.

[MOR12c] M. Moreira, R. Guazzelli, N. Calazans, "Return-to-One DIMS Logic on 4-phase m-of-n Asynchronous Circuits," in IEEE International Conference on Electronics, Circuits, and Systems (ICECS), 2012, pp. 669-672.

[MOR12d] M. Moreira, N. Calazans, "Electrical Characterization of a C-element with LiChEn," in IEEE International Conference on Electronics, Circuits, and Systems (ICECS), PhD Competition Award, 2012, pp. 583-585.

[MOR13a] M. Moreira, C. Menezes, R. Porto, N. Calazans, "Design of NCL Gates with the ASCEnD Flow," in 4th Latin American Symposium on Circuits and Systems (LASCAS), 2013, 6 p.

[MOR13b] M. Moreira, N. Calazans, "Design of Standard cell Libraries for Asynchronous Circuits with the ASCEnD Flow," in IEEE Computer Society Annual Symposium on VLSI (ISVLSI), 2013, 2p.

[MOR13c] M. Moreira, C. Menezes, R. Porto, N. Calazans, "NCL+: Return-to-One Null Convention Logic," in 56th IEEE International Midwest Symposium on Circuits and Systems (MWSCAS), 2013, pp. 836-839.

[MOR13d] M. Moreira, C. Menezes, L. Ost, N. Calazans, "LiChEn: Automated Electrical Characterization of Asynchronous Standard Cell Libraries," in 16th Euromicro Conference on Digital System Design (DSD), 2013, pp. 933-940.

[MOR13e] M. Moreira, F. Magalhães, F. Hessel, N. Calazans, "BaBaNoC: An Asynchronous Network-on-Chip Described in Balsa," in IEEE International Symposium on Rapid System Prototyping (RSP), 2013, pp. 37-43.

[MOR13f] M. Moreira, B. Oliveira, F. Moraes, N. Calazans, "Charge Sharing Aware NCL Gates Design," in 16th IEEE Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2013, pp. 212-217.

[MOR13g] M. Moreira, N. Calazans, "Voltage Scaling on C-elements: A Speed, Power and Energy Efficiency Analysis," in 31st IEEE International Conference on Computer Design (ICCD), 2013, pp. 329-334.

[MOR13h] M. Moreira, F. Moraes, N. Calazans, "Beware the Dynamic C-element," IEEE Transactions on Very Large Scale Integration (VLSI) Systems. Approved for publication as a Brief paper in 2013. Early access.

[MOR14a] M. Moreira, J. Pontes, N. Calazans, "Tradeoffs Between RTO and RTZ in WCHB QDI Asynchronous Design," in International Symposium on Quality Electronic Design (ISQED), 2014. Approved for publication.

[MOR14b] M. Moreira, M. Arendt, R. Guazzelli, N. Calazans, "A New CMOS Topology for Low-Voltage Null Convention Logic Gates Design," in International Symposium on Asynchronous Circuits and Systems (ASYNC), Potsdam, 2014. Approved for publication.

[MOR14c] M. Moreira, N. Calazans, A. Silva, M. Martins, A. Reis, R. Ribas, "Semi-custom NCL Design with Commercial EDA Frameworks: Is it Possible?," in International Symposium on Asynchronous Circuits and Systems (ASYNC), Potsdam, 2014. Accepted for publication.

[MOR14d] M. T. Moreira, R. Guazzelli, G. Heck, N. L. V. Calazans, "Hardening QDI Circuits Against Transient Faults Using Delay-Insensitive Maxterm Synthesis," in Great Lakes Symposium on VLSI (GLSVLSI), Houston, 2014. Accepted for publication.

[MOS14] The MOSIS Service. http://www.mosis.com. 2014.

[MUL57] D. Muller, W. Bartky, "A Theory of Asynchronous Circuits," in International Symposium on the Theory of Switching, Cambridge, 1957, pp. 204-243.

[MYE01] C. Myers, "Asynchronous Circuit Design," New York, John Wiley & Sons, Inc. 2001, 422p.

[NEU13] A. Neutzling, M. Martins, R. Ribas, A. Reis, "Synthesis of threshold logic gates to nanoelectronics," in 26th Symposium on Integrated Circuit and Systems Design (SBCCI), 2013, pp. 1–6.

[NOW11] S. Nowick, M. Singh, "High-performance asynchronous pipelines: an overview," IEEE Design & Test of Computers, 28(5), 2011, pp. 8-22.

[PAR12] F. Parsan, S. Smith, "CMOS implementation of static threshold gates with hysteresis: A new approach," in VLSI and System-on-Chip (VLSI-SoC), Oct 2012, pp. 41–45.

[PAR13] F. Parsan, W. Al-assadi, S. Smith, “Gate mapping automation for asynchronous null convention logic circuits,” IEEE Transactions on Very Large ScaleIntegration (VLSI) Systems, 2013, early access.

[PON10a] J. Pontes, M. Moreira, F. Moraes, N. Calazans, “HERMES-A - An Asynchronous NoC Router with Distributed Routing,” in International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS), Grenoble, 2010, pp. 150-159.

[PON10b] J. Pontes, M. Moreira, F. Moraes, N. Calazans, “Hermes-AA: A 65nm Asynchronous NoC Router with Adaptive Routing,” in IEEE International SoC Conference (SOCC), 2010, pp. 493-498.

[PON12] J. Pontes, N. Calazans, P. Vivet, “Adding Temporal Redundancy to Delay Insensitive Codes to Mitigate Single Event Effects,” in IEEE 18th International Symposium on Asynchronous Circuits and Systems (ASYNC), May 2012, pp. 142–149.

[PRA07] M. Prakash, “Library Characterization and Static Timing Analysis of Asynchronous Circuits,” M.Sc. Thesis, University of Southern California, 2007, 68p.

[RAB03] J. Rabaey, A. Chandrakasan, B. Nikolic, “Digital Integrated Circuits a Design Perspective,” Upper Saddle River, Pearson Education, 2003, 761p.

[RAB13] J. Rabaey, “For Lower Power, Re-Think Computing,” Keynote Speech, Industry Insights - Cadence Community. Available at: http://www.cadence.com/Community /blogs/ii/archive/2012/10/21/jan-rabaey-keynote-for-lower-power-re-think-computing.aspx?postID=1315962. Accessed in 2013.

[REE12] R. Reese, S. Smith, M. Thornton, “Uncle - An RTL Approach to Asynchronous Design,” in IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC), 2012, pp. 65–72.

[SCH11] I. Schwartz, A. Teman, R. Dobkin, A. Fish, “Near-threshold 40nm Supply Feedback C-element,” in 3rd Asia Symposium on Quality Electronic Design (ASQED), 2011, pp. 74-78.

[SHE07] A. Sheibanyrad, I. Panades, A. Greiner, “Systematic Comparison between the Asynchronous and the Multi-Synchronous Implementations of a Network on Chip Architecture,” in Design, Automation & Test in Europe Conference & Exhibition (DATE), Apr 2007, pp. 1–6.

[SIN07] M. Singh, S. Nowick, “MOUSETRAP: High-speed transition-signaling asynchronous pipelines,” IEEE Transactions on Very Large Scale Integration (VLSI), 15(6), Jun 2007, pp. 684–698.

[SMI96] S. Smith, J. Di, “Designing Asynchronous Circuits using NULL Convention Logic (NCL),” Morgan and Claypool Publishers, 2009, 96p.

[SPA01] J. Sparso, S. Furber, “Principles of Asynchronous Circuit Design – A Systems Perspective,” London: Kluwer Academic Publishers, 2001, 337p.

[STE01] K. Stevens, S. Rotem, R. Ginosar, P. Beerel, C. Myers, K. Yun, R. Kol, C. Dike, M. Roncken, “An asynchronous instruction length decoder,” IEEE Journal of Solid-State Circuits, 36(2), 2001, pp. 217–228.

[STE09a] K. Stevens, D. Gebhardt, J. You, Y. Xu, V. Vij, S. Das, K. Desai, “The Future of Formal Methods and GALS Design,” Electronic Notes in Theoretical Computer Science, 245, Aug 2009, pp. 115–134.

[STE09b] K. Stevens, Y. Xu, V. Vij, “Characterization of Asynchronous Templates for Integration into Clocked CAD Flows,” in 15th IEEE Symposium on Asynchronous Circuits and Systems (ASYNC), May 2009, pp. 151–161.

[SUT89] I. Sutherland, “Micropipelines,” Communications of the ACM, 32, 1989, pp. 720-738.

[THO12] Y. Thonnart, E. Beigné, P. Vivet, “A Pseudo-Synchronous Implementation Flow for WCHB QDI Asynchronous Circuits,” in 18th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC), May 2012, pp. 73–80.

[TIE14] Tiempo, http://www.tiempo-ic.com/, 2014.

[TSU12] L. Tsung-Te, J. Rabaey, “Statistical Analysis and Optimization of Asynchronous Digital Circuits,” in 18th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC), 2012, pp. 1-8.

[VER88] T. Verhoeff, “Delay-insensitive codes - an overview,” Distributed Computing, 3(1), 1988, pp. 1-8.

[WEI10] K. Weidong, Z. Peiyi, J. Yuan, R. Demara. “Design of Asynchronous Circuits for High Soft Error Tolerance in Deep Submicrometer CMOS Circuits,” in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 18(3), 2010, pp. 410-422.

[XUG10] G. Xuguang, L. Yu, Y. Yintang. “Performance Analysis of Low Power Null Convention Logic Units with Power Cutoff,” in Asia-Pacific Conference on Wearable Computing Syst. (APWCS), 2010, pp. 55-58.

[YAH06] E. Yahyah, M. Renaudin, “QDI Latches Characteristics and Asynchronous Linear-Pipeline Performance Analysis,” TIMA Lab, Research Reports, 2006.

[YAN10] S. Yancey, S. Smith, “A differential design for C-elements and NCL gates,” in 53rd IEEE International Midwest Symposium on Circuits and Systems (MWSCAS), 2010, pp. 632–635.

[YAN11] Y. Yang, Y. Yang, Z. Zhu, D. Zhou, “A high-speed asynchronous array multiplier based on multi-threshold semi-static NULL convention logic pipeline,” in IEEE International Conference on ASIC (ASICON), 2011, pp. 633-636.

[ZIE07] A. Ziesemer, C. Lazzari, R. Reis, “Transistor level automatic layout generator for non-complementary CMOS cells,” in IFIP International Conference on Very Large Scale Integration (VLSI-SoC), 2007, pp. 116 –121.

[ZIE14a] A. Ziesemer, R. Reis, M. Moreira, M. Arendt, N. Calazans, “Automatic Layout Synthesis with ASTRAN Applied to Asynchronous Cells,” in IEEE Latin American Symposium on Circuits and Systems (LASCAS), 2014. Accepted for publication.

[ZIE14b] A. Ziesemer, R. Reis, M. Moreira, M. Arendt, N. Calazans, “A Design Flow for Physical Synthesis of Digital Cells with ASTRAN,” in Great Lakes Symposium on VLSI (GLSVLSI), Houston, 2014. Accepted for publication.